| Exp. No: 1 | Installation of VirtualBox with Linux OS |
|---|---|
| **Date:** 24/8/2020 | |

**Aim:**

To Install VirtualBox with different flavours of Linux OS on top of windows7 or 8 or 10 OS.

**Procedure:**

The installation is divided into

1. Installation of VirtualBox on Windows 10/8/7
2. Creation of Ubuntu (Linux) VM
3. Installation of Linux OS on VirtualBox

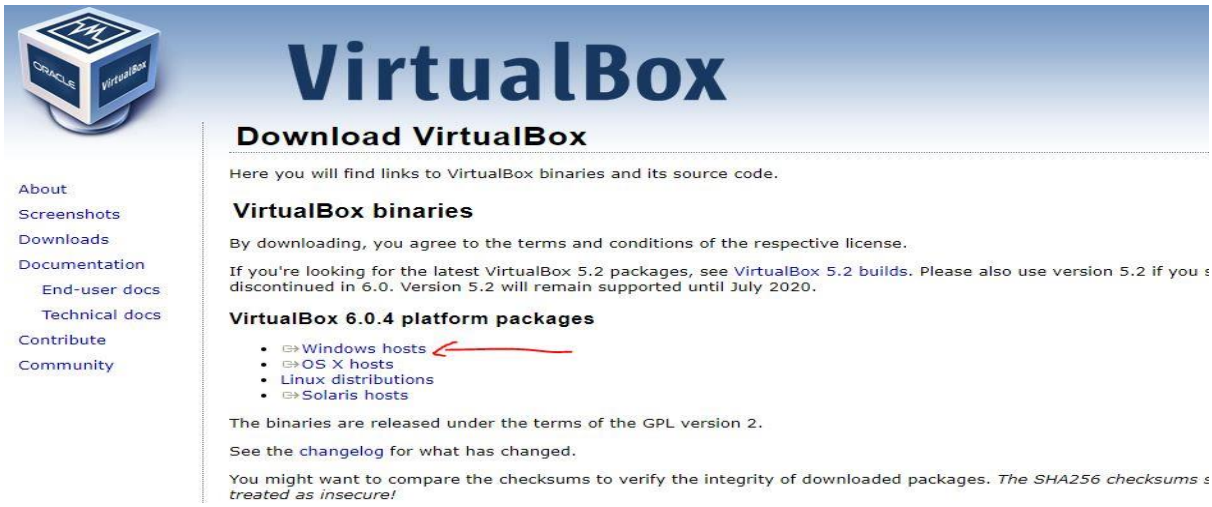
**Installation of VirtualBox on Windows 10/8/7**

1. **Download** VirtualBox software from **Oracle official website.**
2. **Double-click** on downloaded **VirtualBox** Win.exe file to bring up the welcome screen. Click **Next**.
3. Installation files and set the installation path. If you are not familiar, then keep the default configuration, select the **Next** button.
4. Leave the pre-selected **VirtualBox** shortcuts as it is and click on **Next** button.
5. When installing VirtualBox, it involves network functions. The wizard will automatically create a **virtual network card,** which will temporarily interrupt your network. But of course, it will return to normal immediately. So, click **Yes**.
6. Now you can go to install this virtualization software. Click **Install**.During the period, you can see that the current network was interrupted and immediately resumed.
7. Click **Finish** to launch Oracle VM VirtualBox.

**Screenshots of the above steps:**

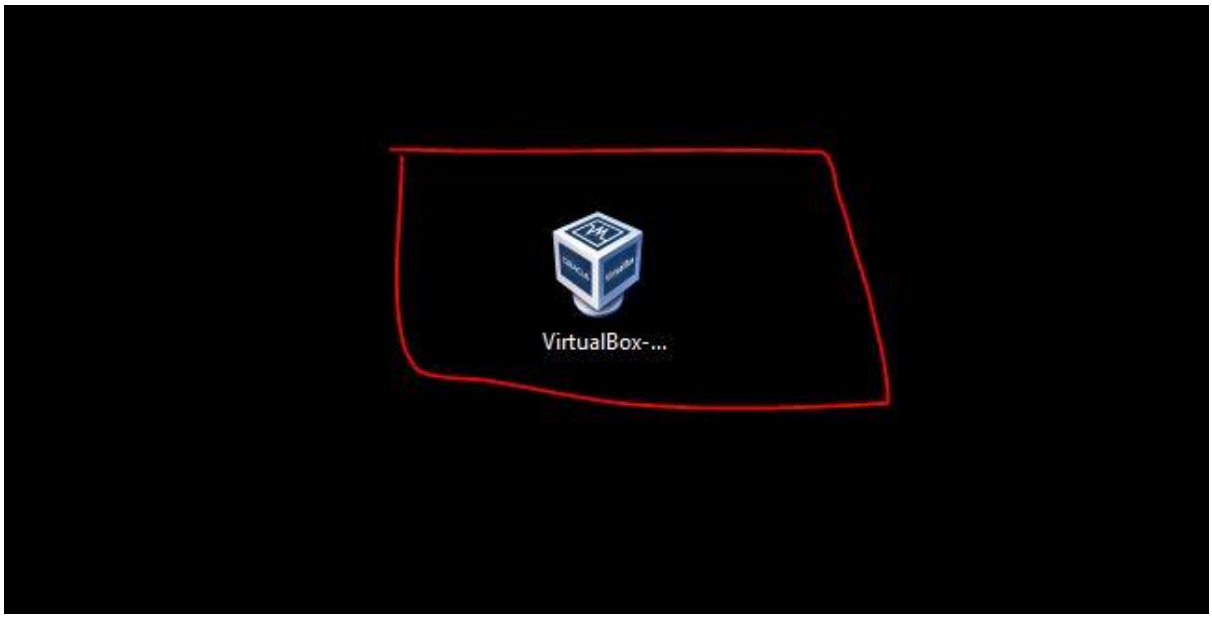
**Step 1: Download VirtualBox for Windows 10/8/7**

**Download** VirtualBox software from **Oracle official website**: **Download VirtualBox**

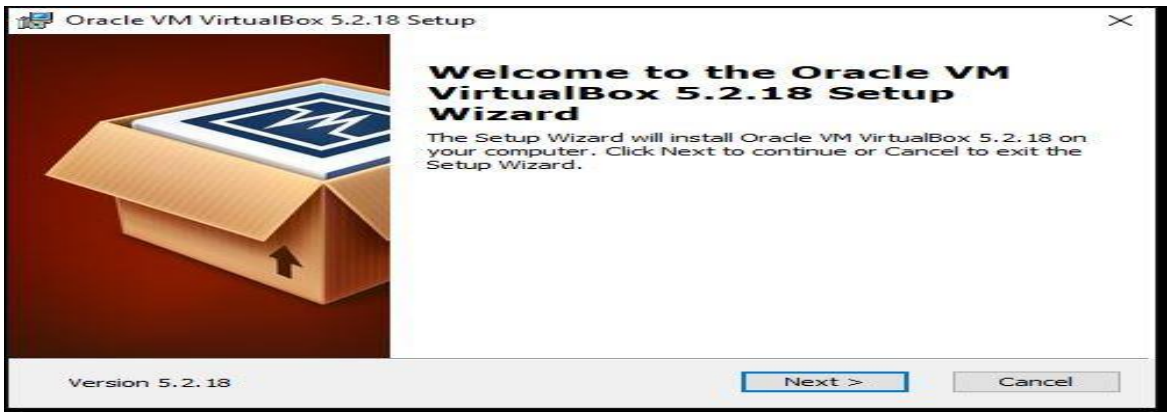



**Step 2: Run the VirtualBox.exe file**

The downloaded VirtualBox file will be in EXE format to run that just double click on it and run it as administrator.




Click on **Next** button to start Oracle VirtualBox installation Setup Wizard.

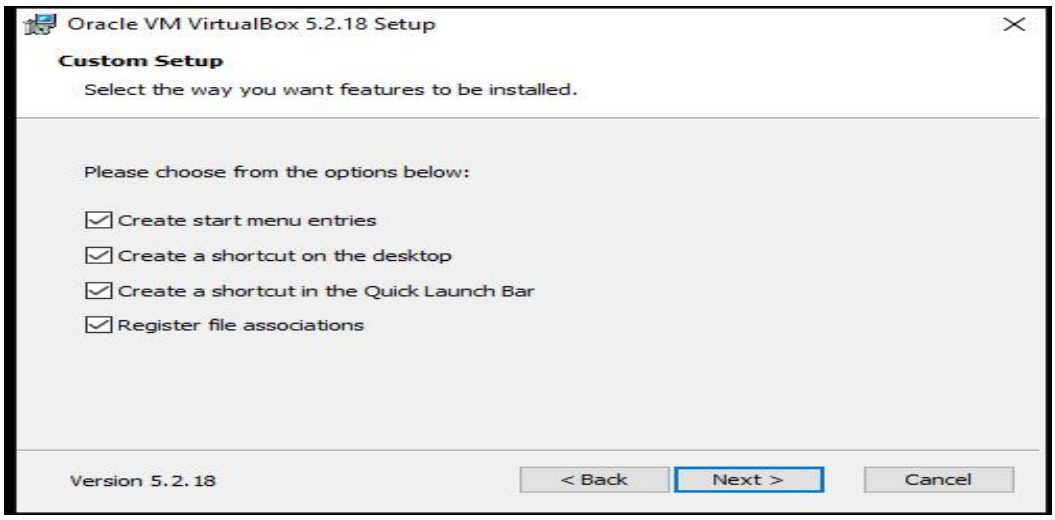


**Step 3: VirtualBox shortcuts**

At this stage, you will see multiple shortcuts:

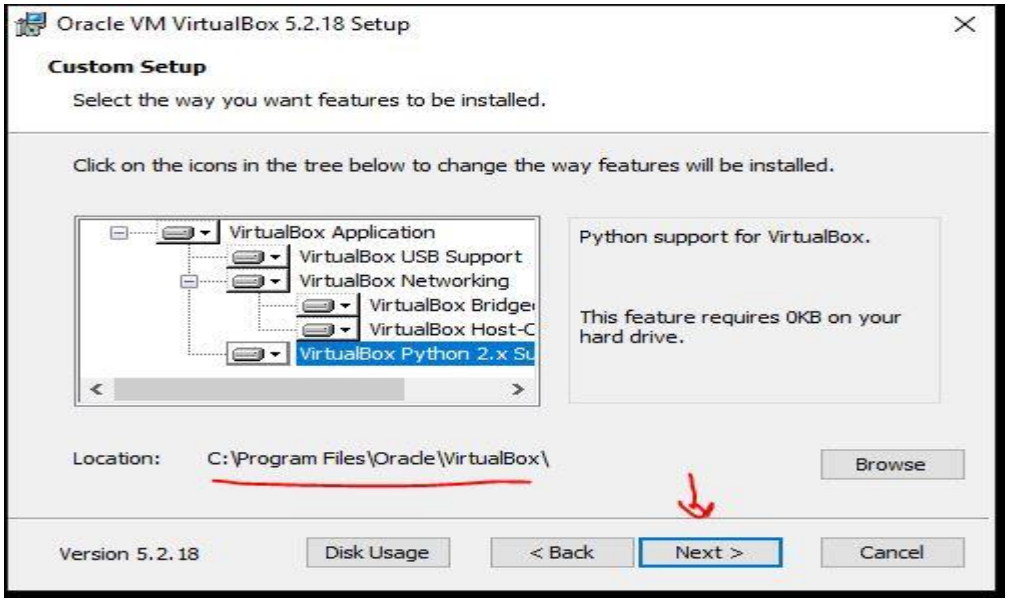
- **Create start menu entries:** To create a Virtualbox shortcut in the start menu of the Windows 10/8/7
- **Create a shortcut on the desktop**: This will create a shortcut on Desktop
- **Create a shortcut in the Quick Launch Bar**: You will get a shortcut in the Taskbar.
- **Register file associations**: Create Virtualbox file entries in Windows registries.

Leave them as it is and click on the **NEXT** button.



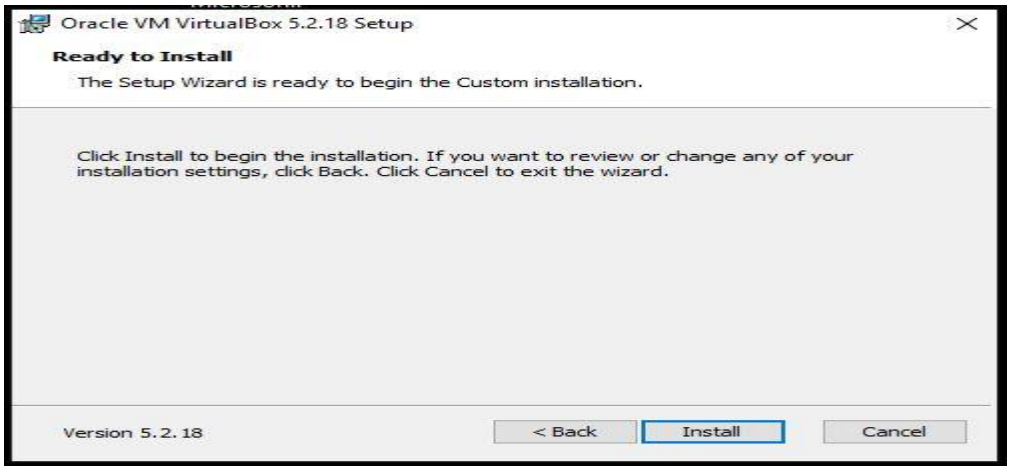

**Step 4: File Location**

By default the VirtualBox will install its core files in the C: Drive. In case you have low space on the C: Drive, then just click on the Browse button and select the location where you want to install it. However, if you are not acquainted with this option then simply leave it as default and click on **NEXT** button.



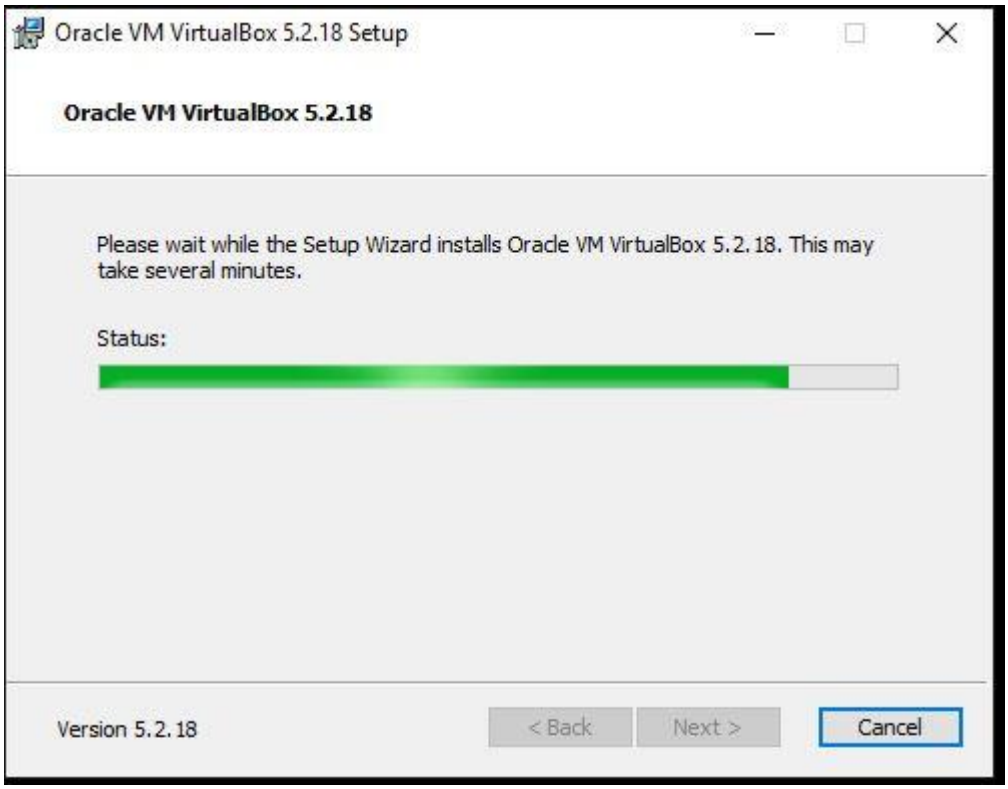

**Step 5: Install VirtualBox**

Click on the **Install** button to begin the installation.



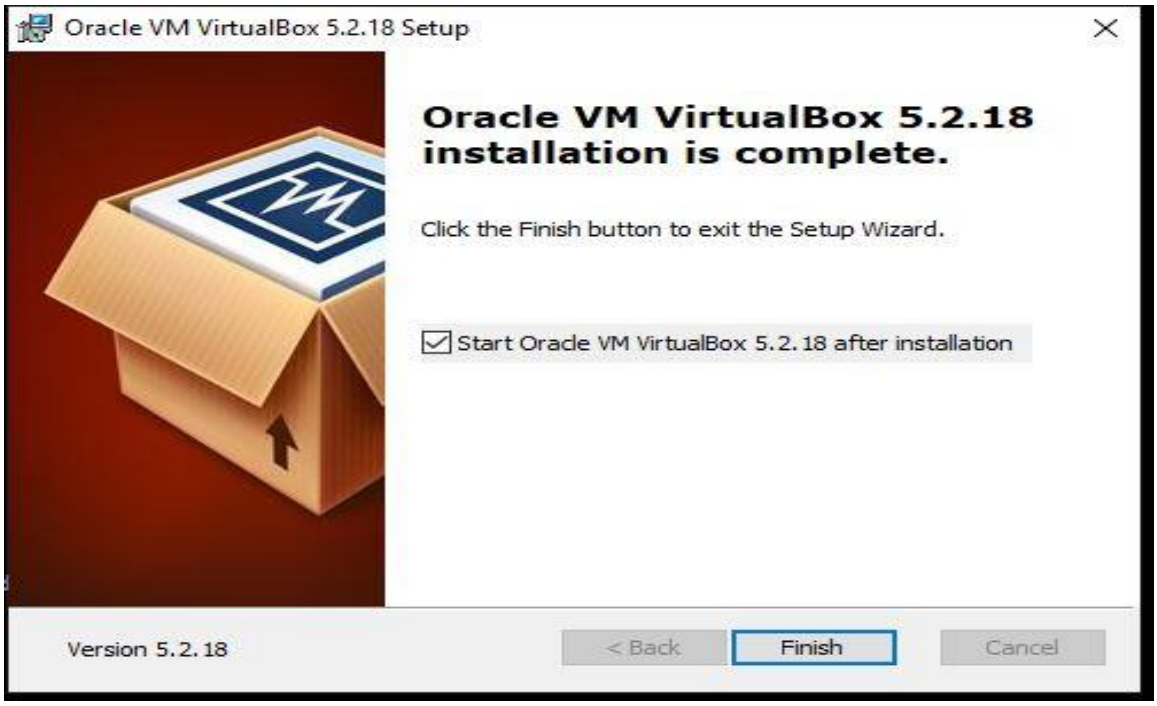

**Step 6: Warning: Network Interfaces**

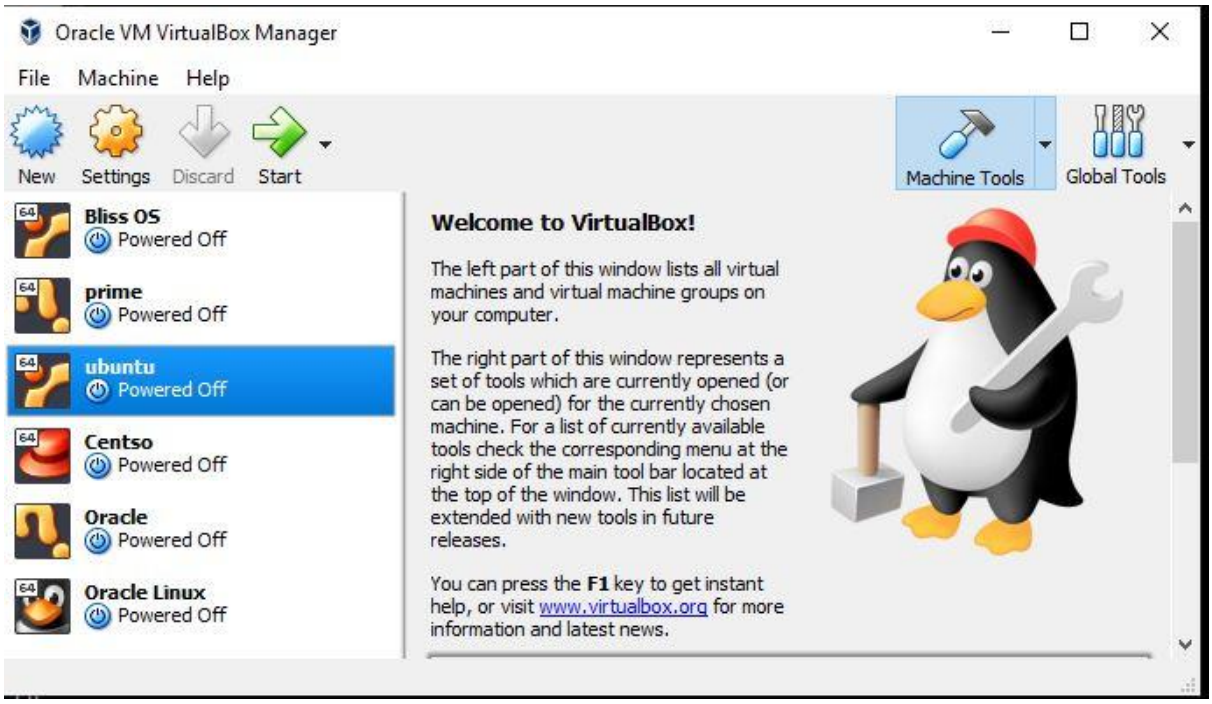
To create Virtual Adapters, the VirtualBox will reset your network connection and disconnect it temporarily for a few seconds and then again it will return to its normal state. So, click on the **YES** button.




**Step 7: Installation is completed**

After installing, the installation wizard will show you a **Finish** button, click on that and it will start the VirtualBox on your Windows 10/7/8 machines.



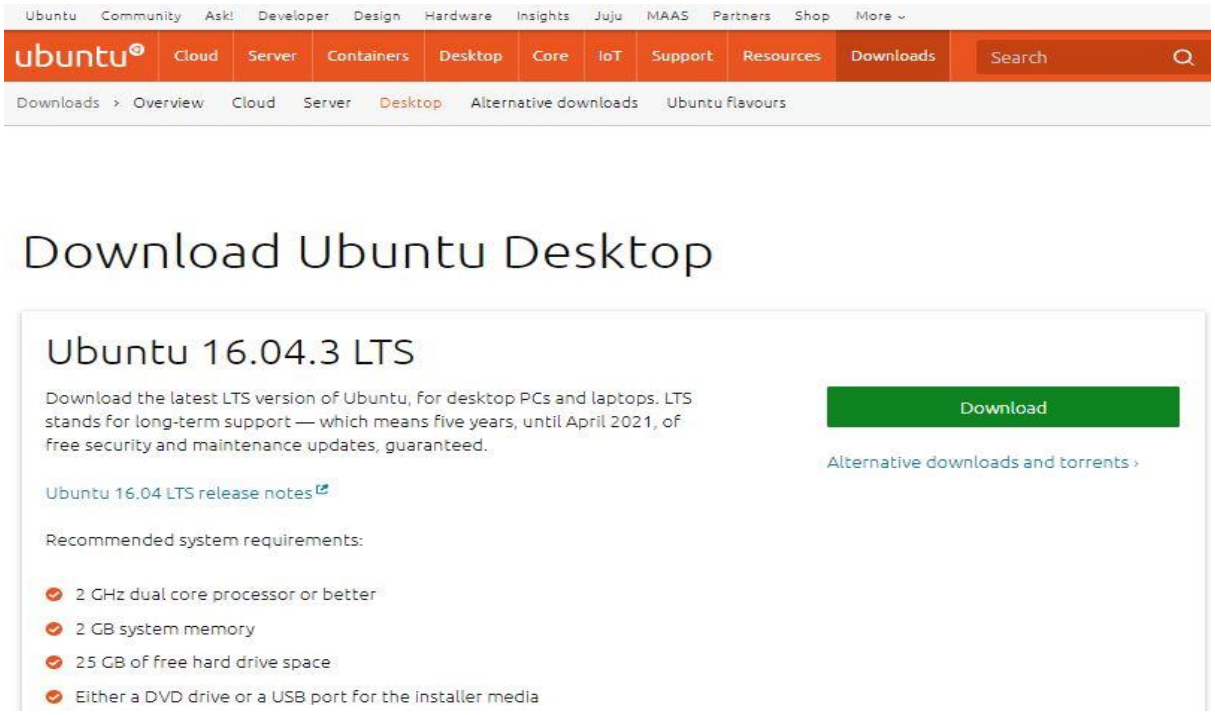



## Creation of Ubuntu VM in VirtualBox

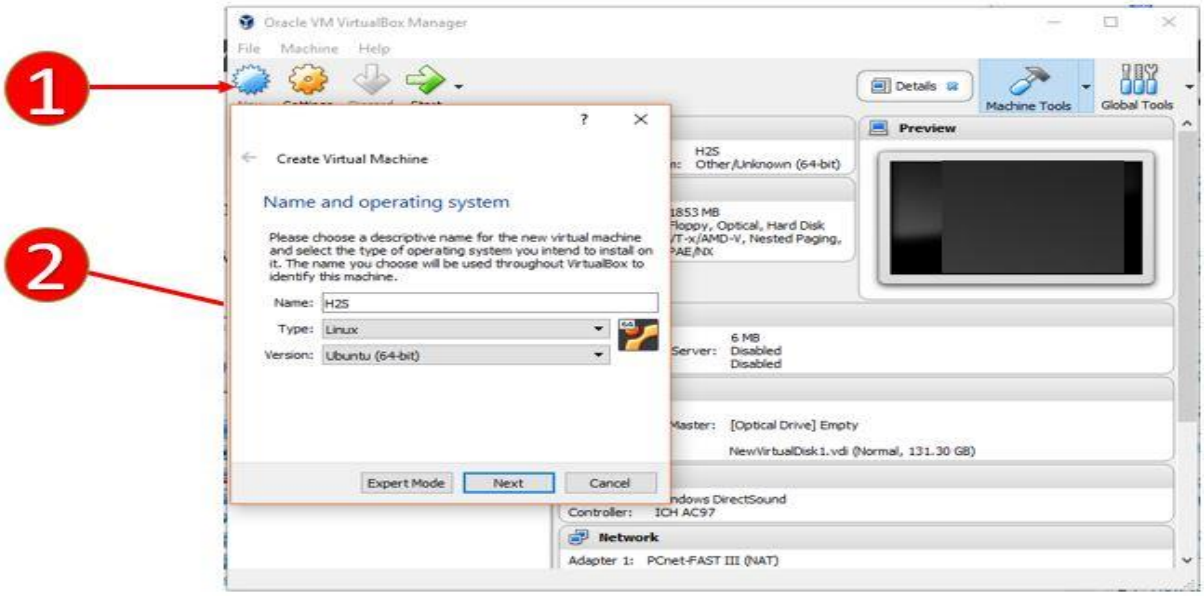
Step 1: Download Ubuntu OS

The open source Ubuntu Linux comes in different flavors and you can download any of them from the official Ubuntu's website. Here is the Link: www.ubuntu.com/download/desktop.

Note: If you already have the Ubuntu.iso file then leave this step.

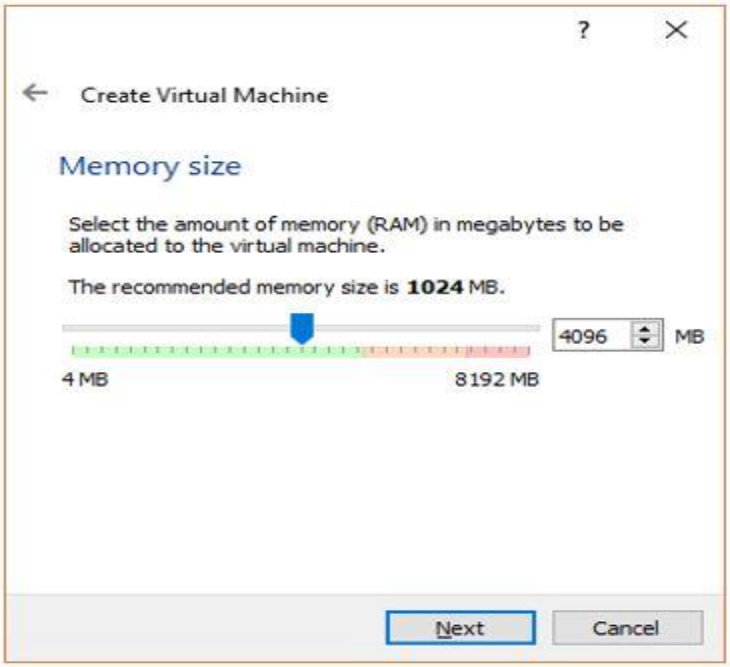


**Step 2:** After successful Virtualbox installation, run it to create an **Ubuntu VM**. Click on **New** and give some name to your Ubuntu VM. For example, here we have used **H2S**. From the type drop-down box select the OS type which is  Linux and Version is Ubuntu 64 bit. If you have Ubuntu 32 bit version then please select that.



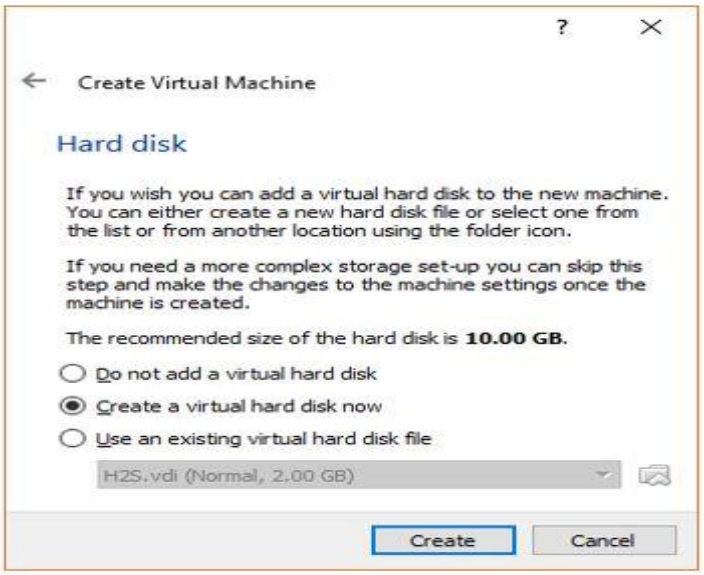

**Step 3: How Much Memory Do You Give Your Virtual Machine**

In this step, the Vitrualbox will ask to set the Virtual Machine Memory Size for Ubuntu VM. The recommended RAM for Ubuntu OS is 2 GB or 2048 MB but you can assign more for better performance. For example here in our Windows 10 PC, we have maximum 8GB memory and out of that, we are going to assign 4GB to Ubuntu VM.



**Step 4: Create A Virtual Hard Drive For Ubuntu VM (Virtual Machine)**

After assigning the memory, its time to provide some space for the installation of Ubuntu VM. To create a new virtual hard disk select the option "**Create a virtual hard disk now**" and click "Create".




**Step 5: Choose Virtual Hard disk Type**

The Virtualbox offers three type of Virtual hard drives:

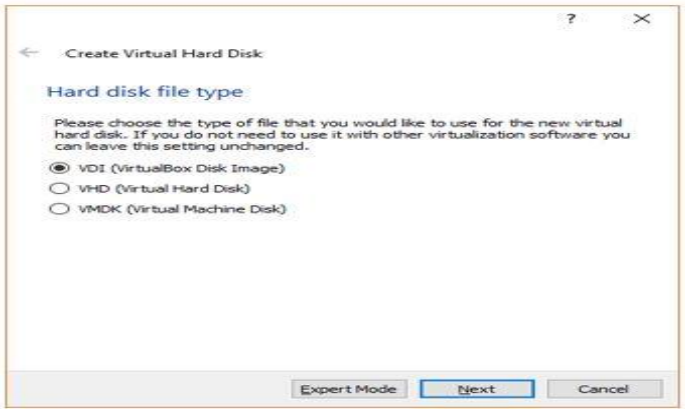
1.VDI- Virtual Disk Image

2.VHD- Virtual Hard Disk

3.VMDK- Virtual Machine Disk

If you are planning to use the Virtual hard drive with some other virtualization software in future such as with VM player or Windows Hyper-V then you choose according to that otherwise leave it as it is "***VDI***" and click on ***NEXT***.

There are a number of different hard drive types that you can choose from. Choose "VDI" and click "Next".



**Step 6: Storage on Physical Hard disk for Ubuntu VM**

To install Ubuntu Virtual Machine files on physical storage of Windows 10, the Virtualbox offers two options:

1. Dynamically allocated
2. Fixed Size

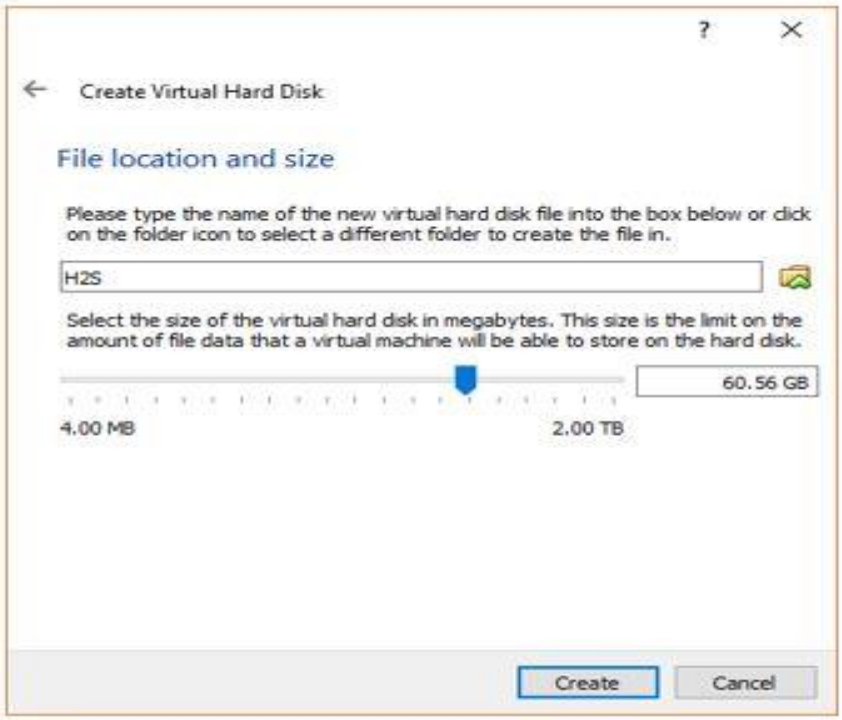
The Dynamically allocated hard disk option will only use space as it required. For example, you assigned 30 GB to Ubuntu VM but if it requires 10Gb initially then the Virtualbox uses only that and not going to block the whole 30GB. And in future, it requires more, expands according to that. It is good in terms of disk space but not performance wise.

Fixed Size on another hand block whole space you have assigned to the VM. For example, if you allocated the 30GB, then the machine will straight away assign that portion from the physical hard drive to Ubuntu VM. The Fixed size allocation is better for performance but take some time to create if you are assigning a large amount of space.

Choose the option you would like and click "Next".

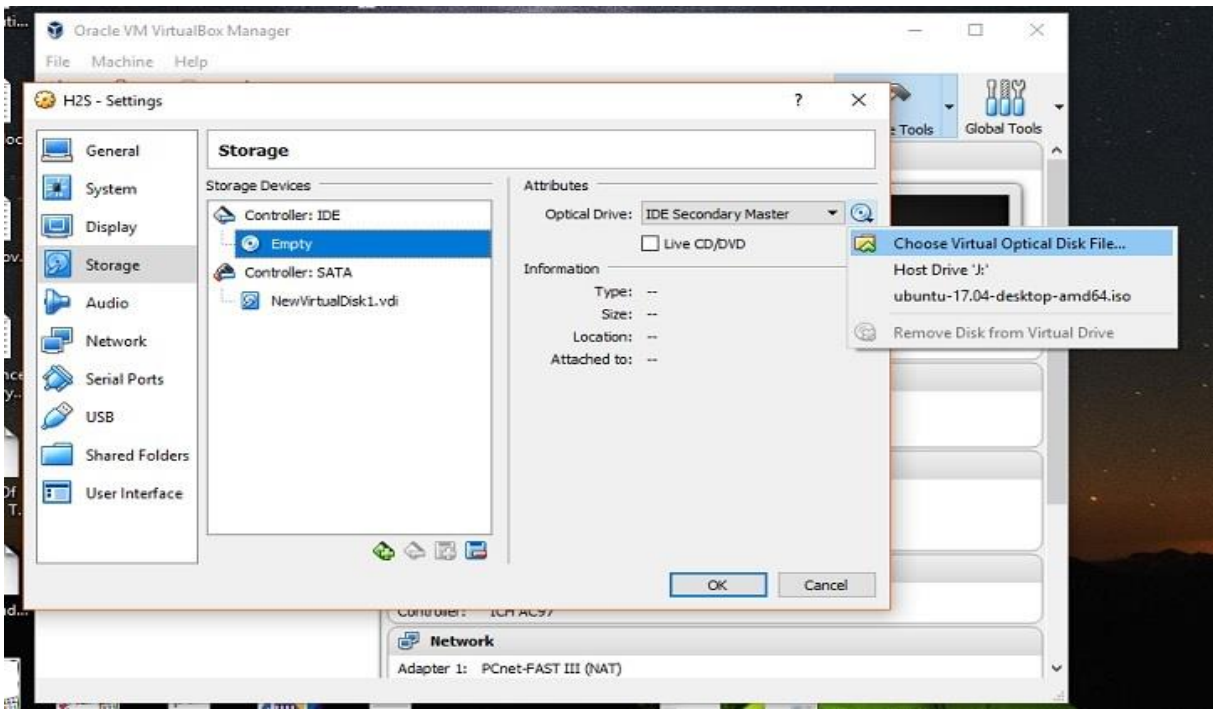

**Step 7: Virtual Hardrive File location and Size**

Give some name to your virtual hard disk and select the amount of space you want to assign the Ubuntu VM. The minimum recommended space is 25GB. You can assign more for better performance.




**Step 8: Assign Ubuntu ISO to VirtualBox**

Go to setting and from storage click on the empty CD-Rom icon and from the Optical drive option **choose the Virtual optical Disk File** and select the Ubuntu.iso file which is our downloaded beginning of this article. After selecting the ISO file click **OK**.

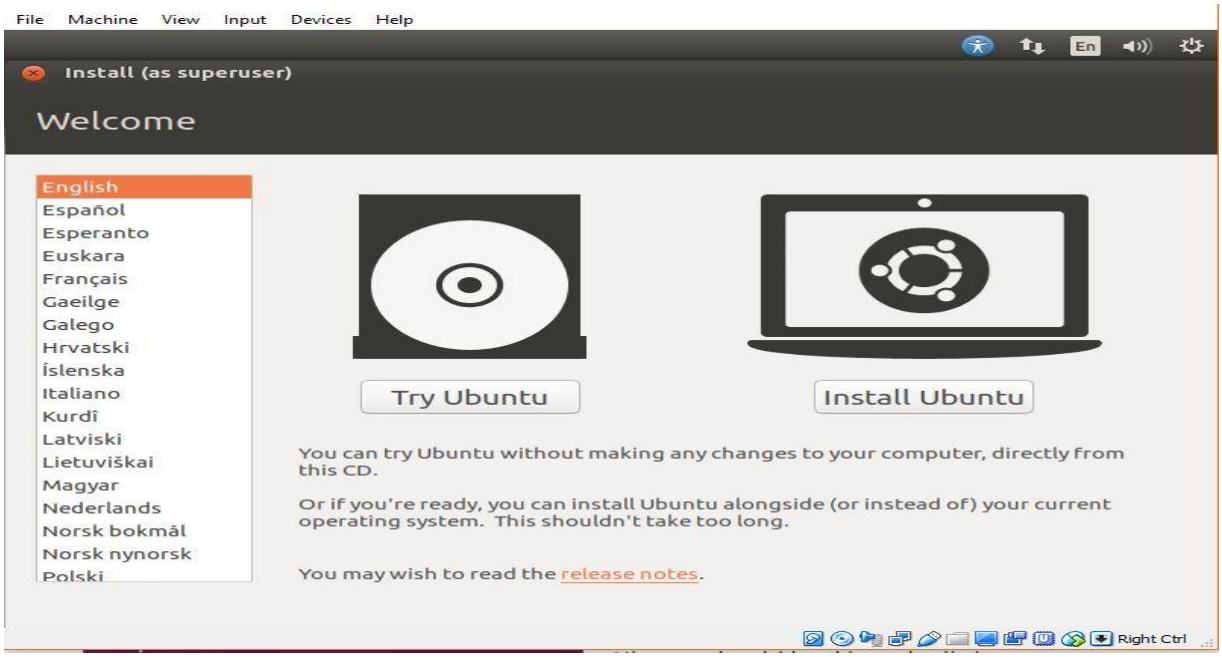


## Installation of Ubuntu OS on VirtualBox

**Step 1:** On the top of the Virtual box you will an option "**START**", click on that to initialize the Ubuntu installation process on Windows 10.

**Step 2:** The Ubuntu first screen will load two options **Try Ubuntu** or **Install Ubuntu.**

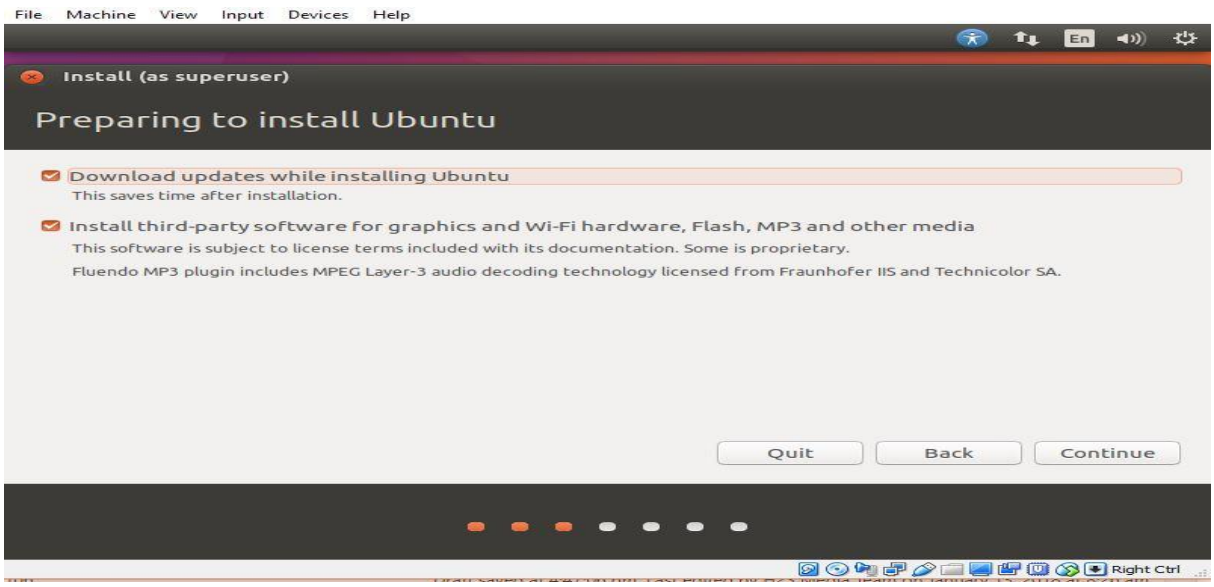
Select the installation language and after that the "**Install Ubuntu**" option.




**Step 3:** If you have enough internet bandwidth and then you can select the option download the updates while installing Ubuntu.

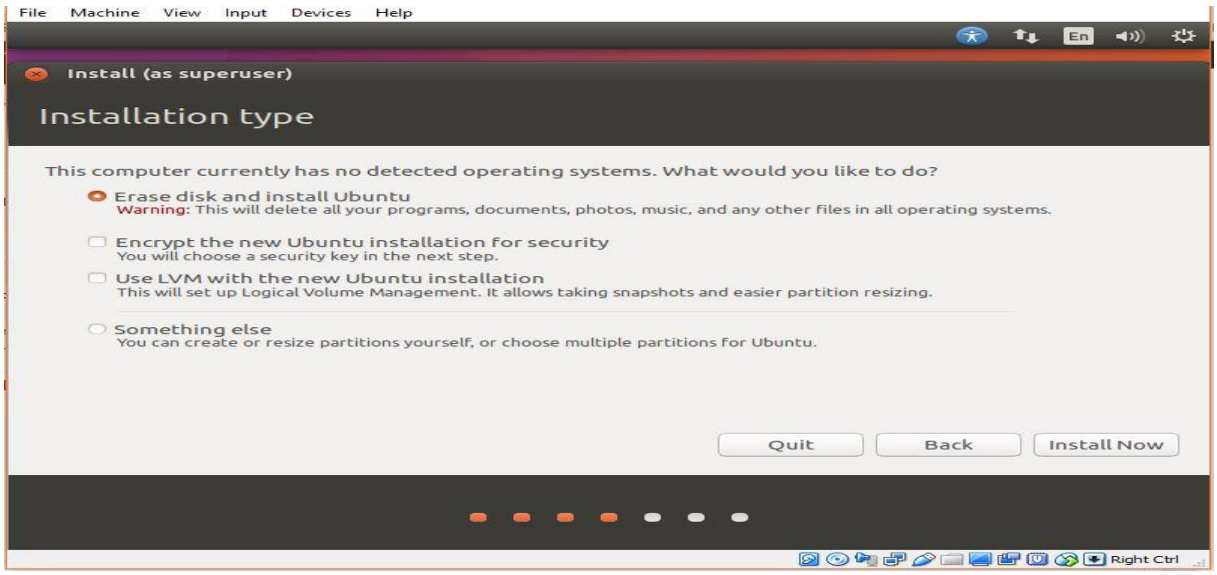
The second option doesn't require an internet connection and recommended to select it to install third party software such as graphics driver, Mp3 player, flash and other media files.

Click "Continue".

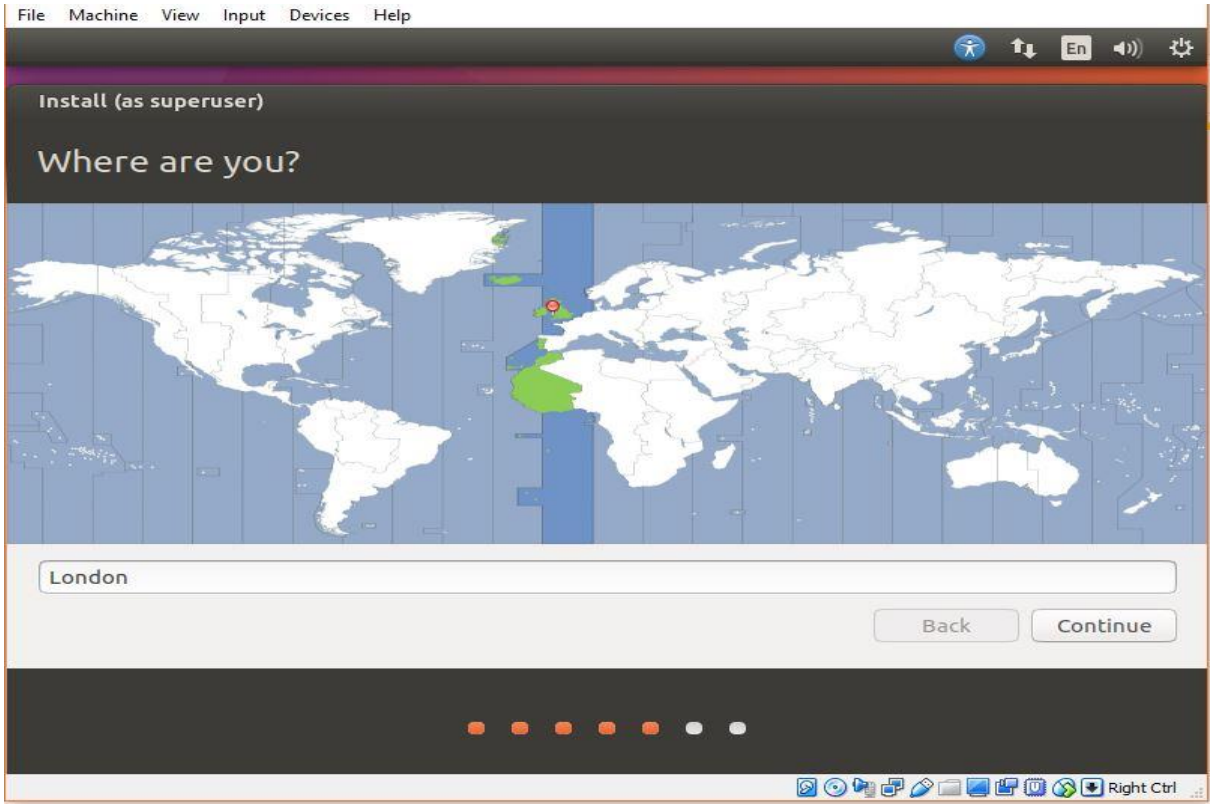


**Step 4:** In this step, you will decide how you want to decide the Ubuntu either clean installation or dual boot with some other OS. Leave the default option the "Erase disk and install Ubuntu" option because it is on the virtual machine won't going to affect the physical Windows 10 machine.

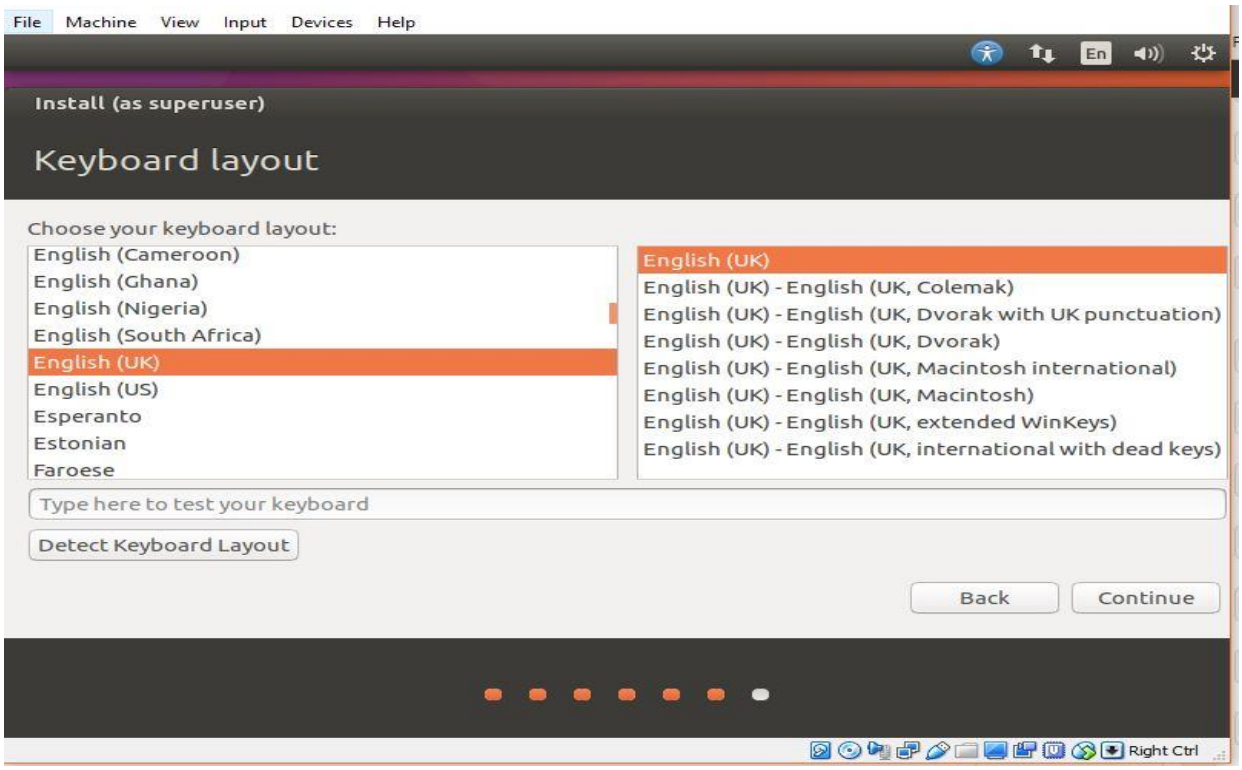
Click "***Install Now***" and "***Continue***".




**Step 5:** Select your country to sync the Ubuntu OS time zone with your's and click "Continue".




**Step 6:** Click on the "**Detect Keyboard Layout**" to automatically detect your keyboard layout and if the machine not able to do it, you can select it manually. Click "Continue".




**Step 7:** Create a user and set the password for your Virtual Ubuntu machine and click on continue to install the Ubuntu Virtualbox.


Finally, the Ubuntu is installed on **VirtualBox on Windows 10** as host machine

**Result:**

Thus, the Installation of VirtualBox with different flavours of Linux OS on top of windows7 or 8 or 10 OS completed successfully.

| Exp. No: 2 | Installation of C Compiler in VirtualBox |
|---|---|
| **Date:** 31/08/2020 | |

**Aim:**

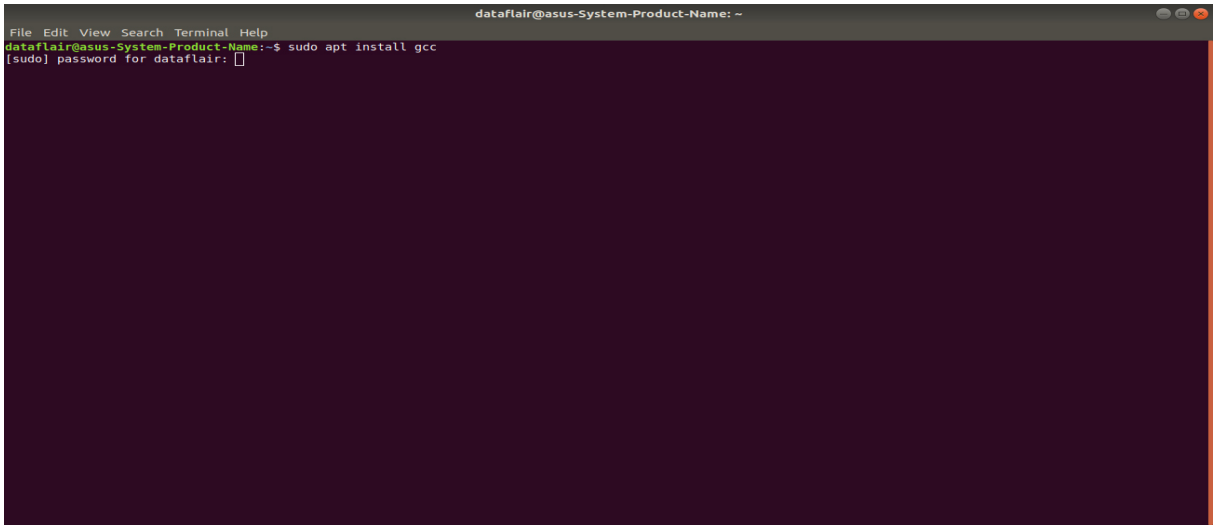
To Install a C compiler in the virtual machine created using virtual box and execute Simple Programs.

**Procedure:**

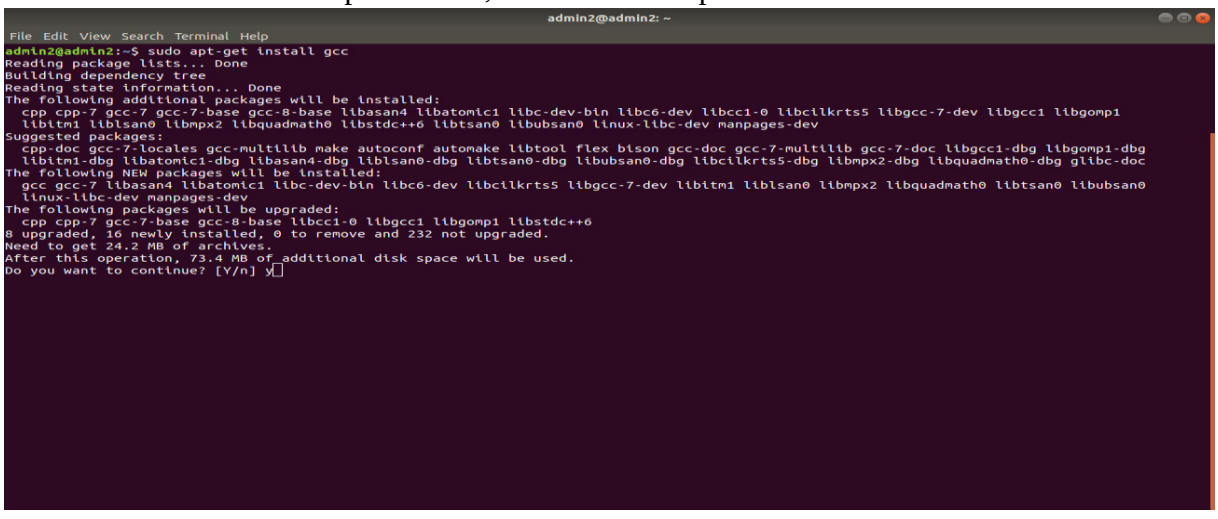
1. Open the VirtualBox application & then Start the Ubuntu Virtual Machine installed.
2. Open the Terminal Command prompt by clicking terminal icon in desktop or using shortcut **Ctrl + Alt + T**
3. Enter the following command in the terminal window

   *sudo apt install GCC*

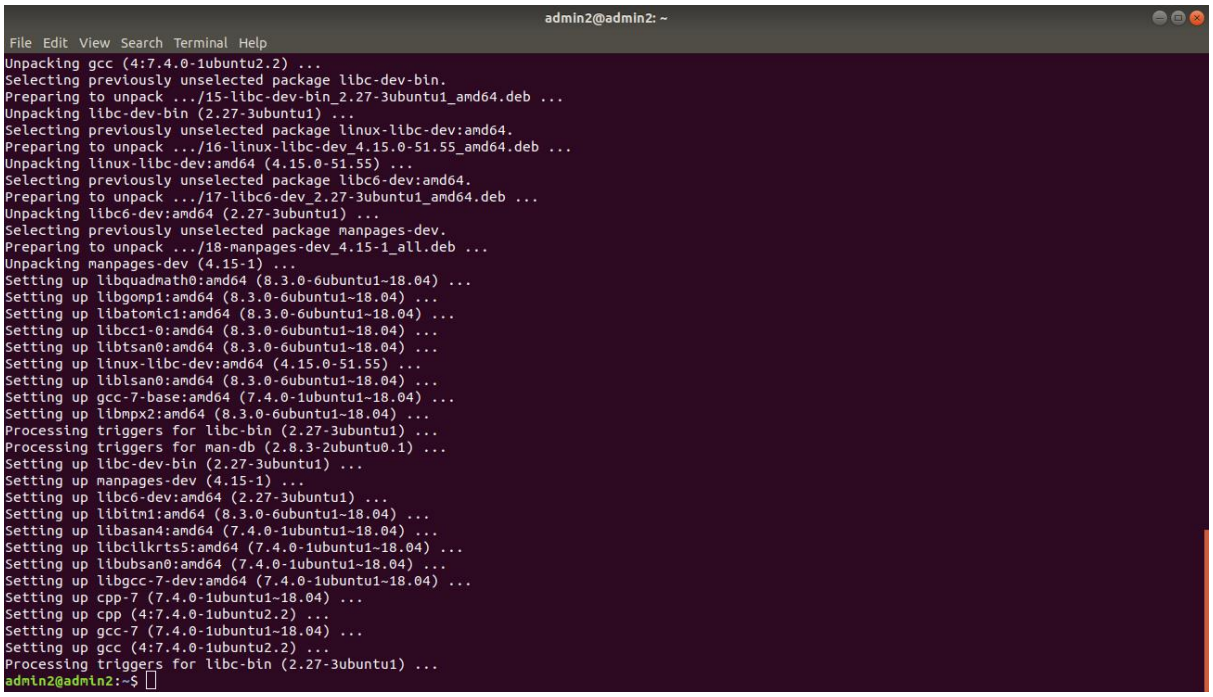
   Here, GCC is the C Compiler. Enter admin password if prompted.




4. If we have the installation permission, the installation proceeds as follows

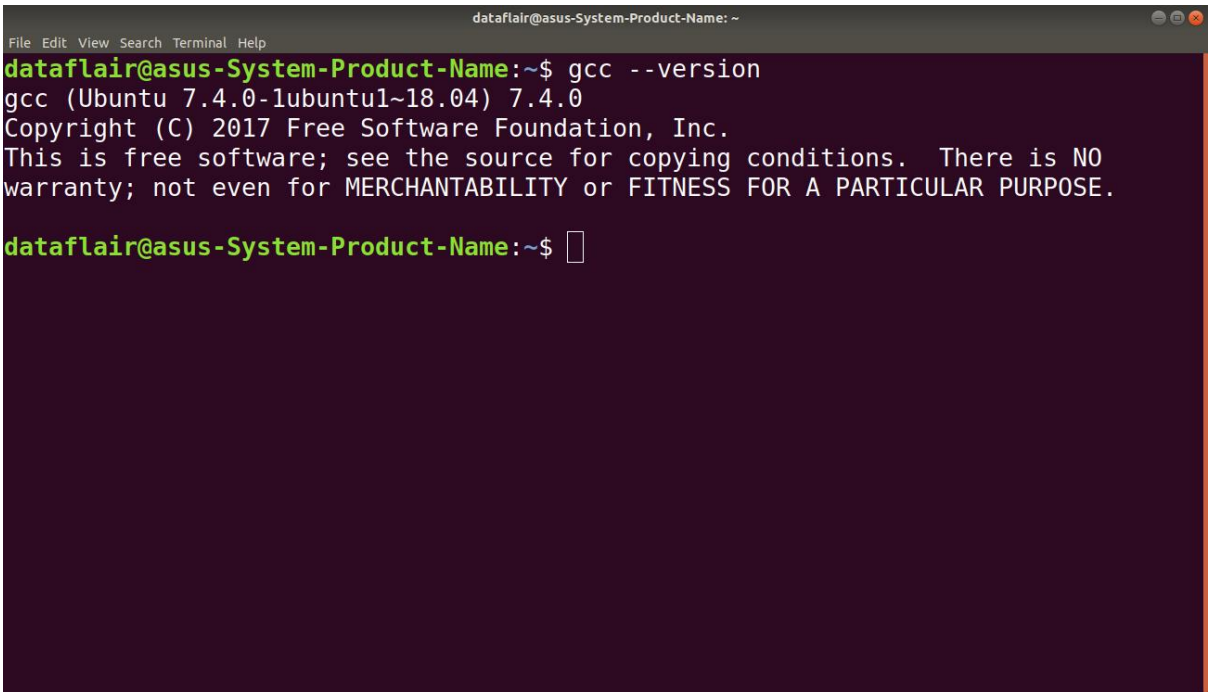


5. Type ‘y’ when the command prompt asks “Do you want to continue?” and then press Enter to continue the installation.

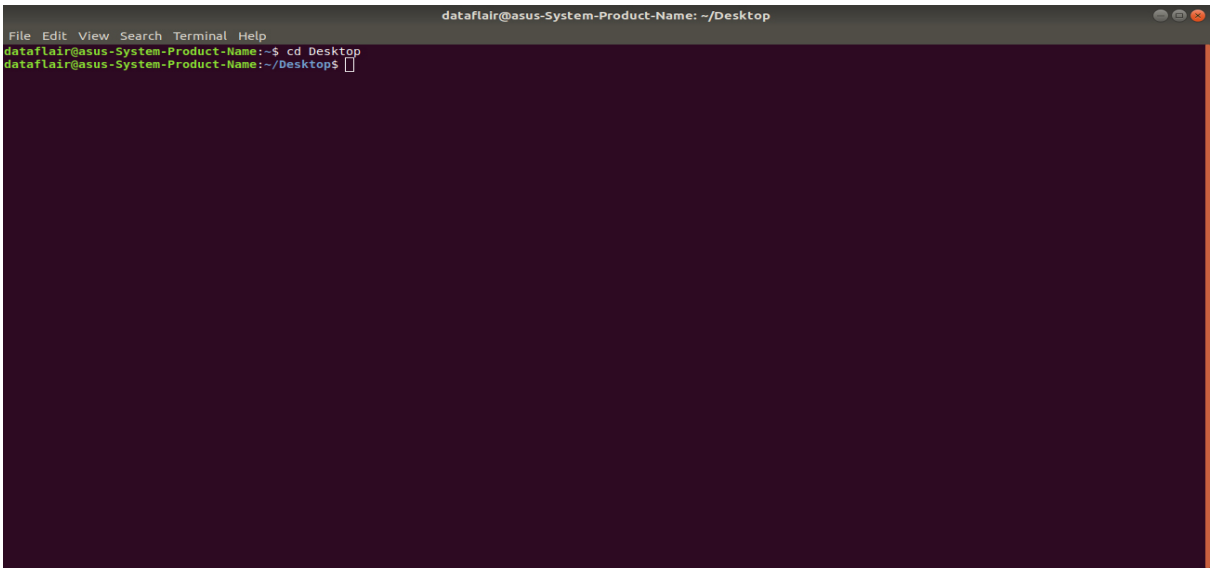



6. After successful installation, Verify the installation by checking the version number of gcc using following command.

**GCC — version**

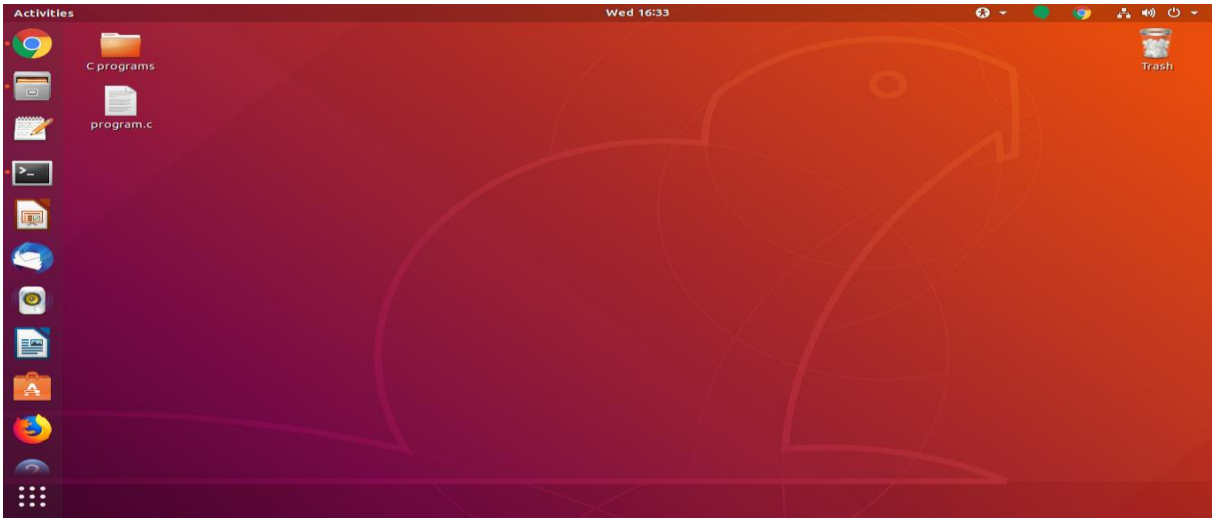



7. In terminal, Move the desired directory (Ex: Desktop) where you want to save the program. **cd Desktop**

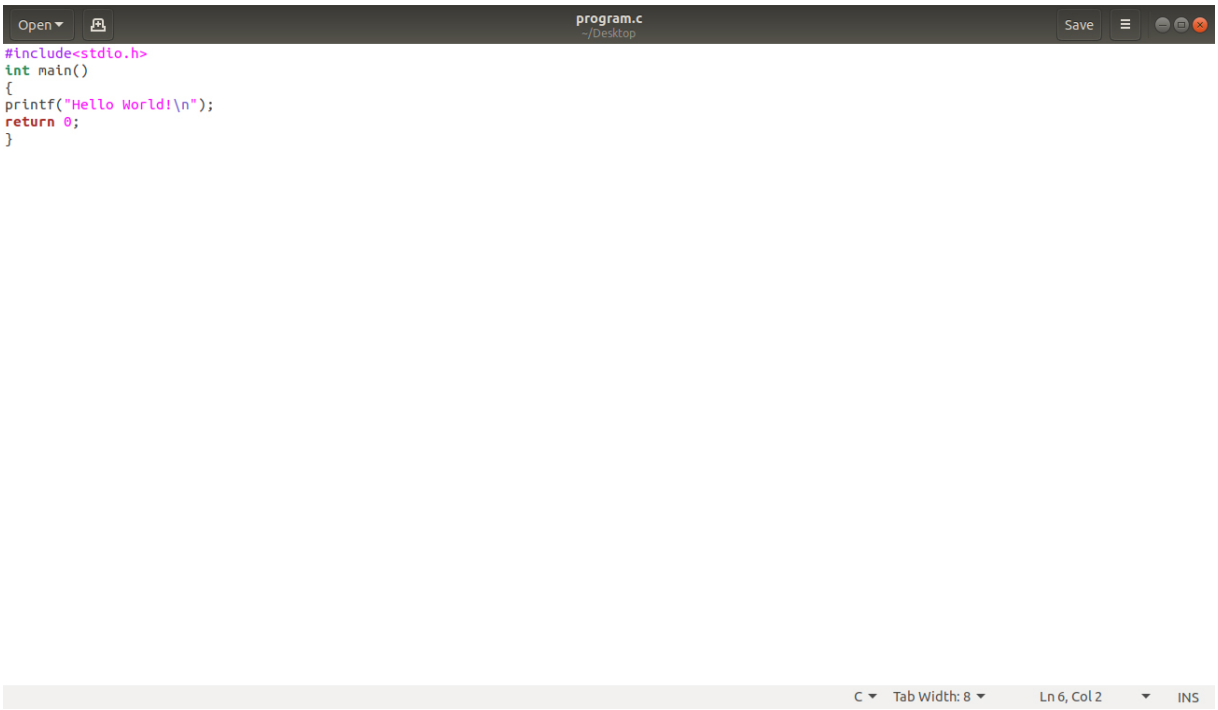



8. The command for creating a program in C is: **touch program.c**

   Now, a file has been created in our Desktop folder called program.c

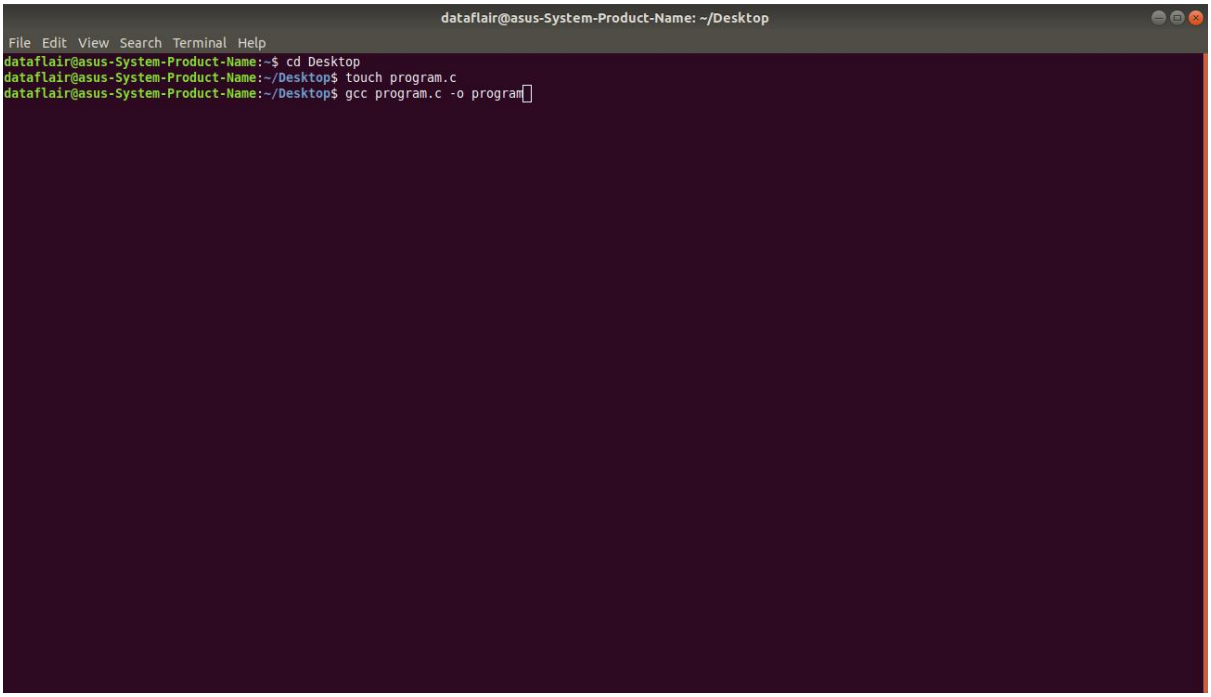



8. Open this file and write a basic code – **"Hello World!"** or any code required.

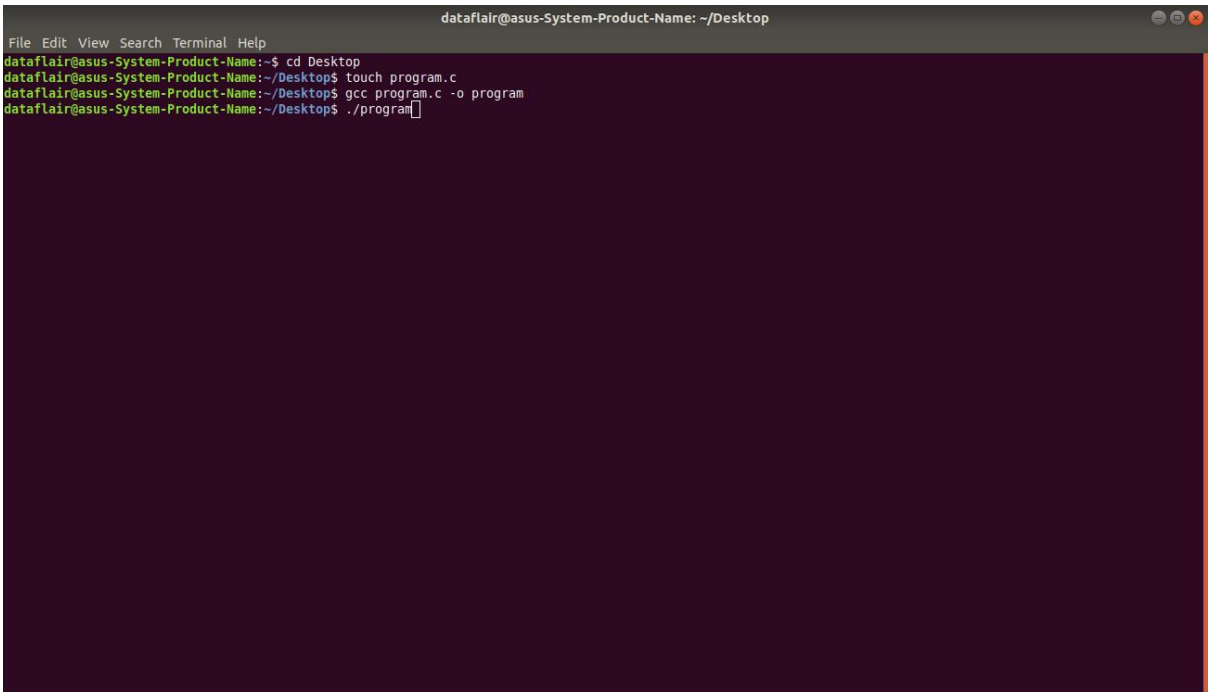



9. To compile the code, we use the GCC command:

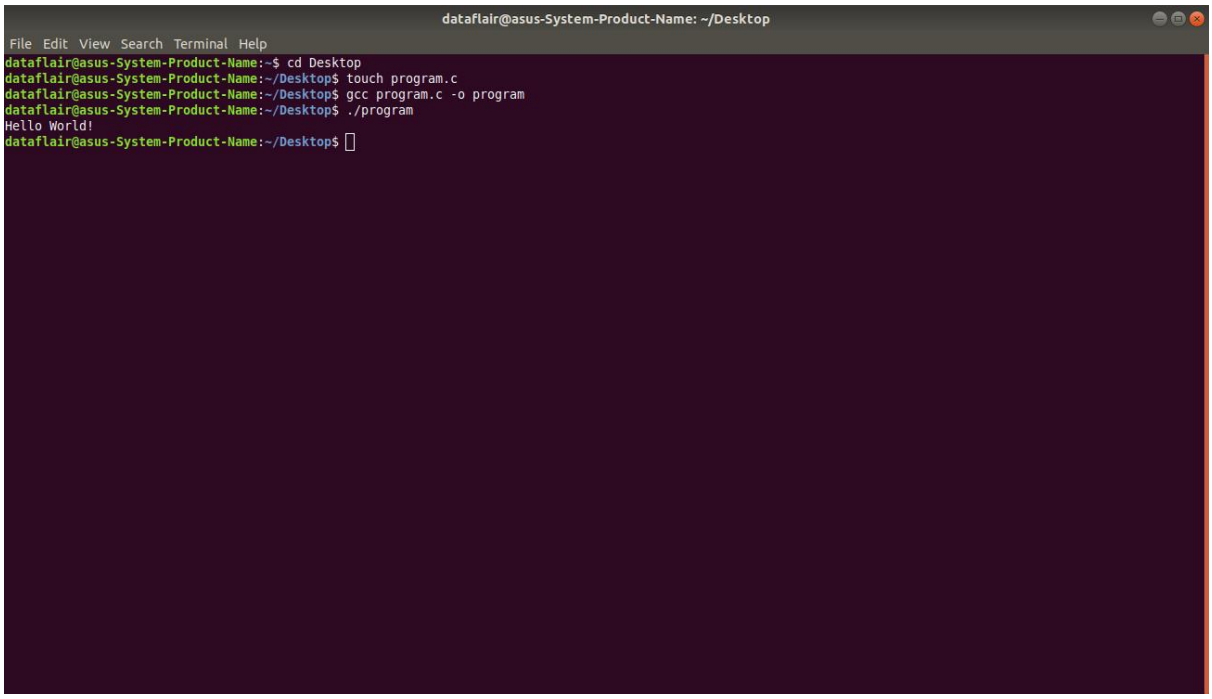
**GCC program.c -o program**




10. To run the program, use the command: **./program**




11.On successful execution, the output will look like




**Result:**

Thus, the installation of C Compiler & execution of C Program is completed successfully.

| Exp. No: 3 | Installation of Google App Engine & Create “Hello world” App |
|---|---|
| **Date:** 07/9/2020 | |

**Aim:**

To Install Google App Engine and Create hello world app using python.

**Procedure:**

1. Install notpad++ editor or any other editor to write program.
2. Download latest version of Python for windows and Install it from

**https://www.python.org/downloads/**

**3.** Download and install Google App Engine from
**https://cloud.google.com/appengine/downloads?csw=1**
(Download the Cloud SDK installer for windows. )

4. Set the environment variable path to point to python & GAE installation directories as follows

Path= C:\Program Files\Python39\; C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin

5. Write a program for hello world in notpad++ editor as follows ( the indentations in each line of the code is important)

**Program 1: test.py**

```
import webapp2

class MainPage(webapp2.RequestHandler):
    def get(self):
        self.response.write("Hello World")

app=webapp2.WSGIApplication([('/', MainPage), ], debug=True)
```

**Program 2: app.yaml**

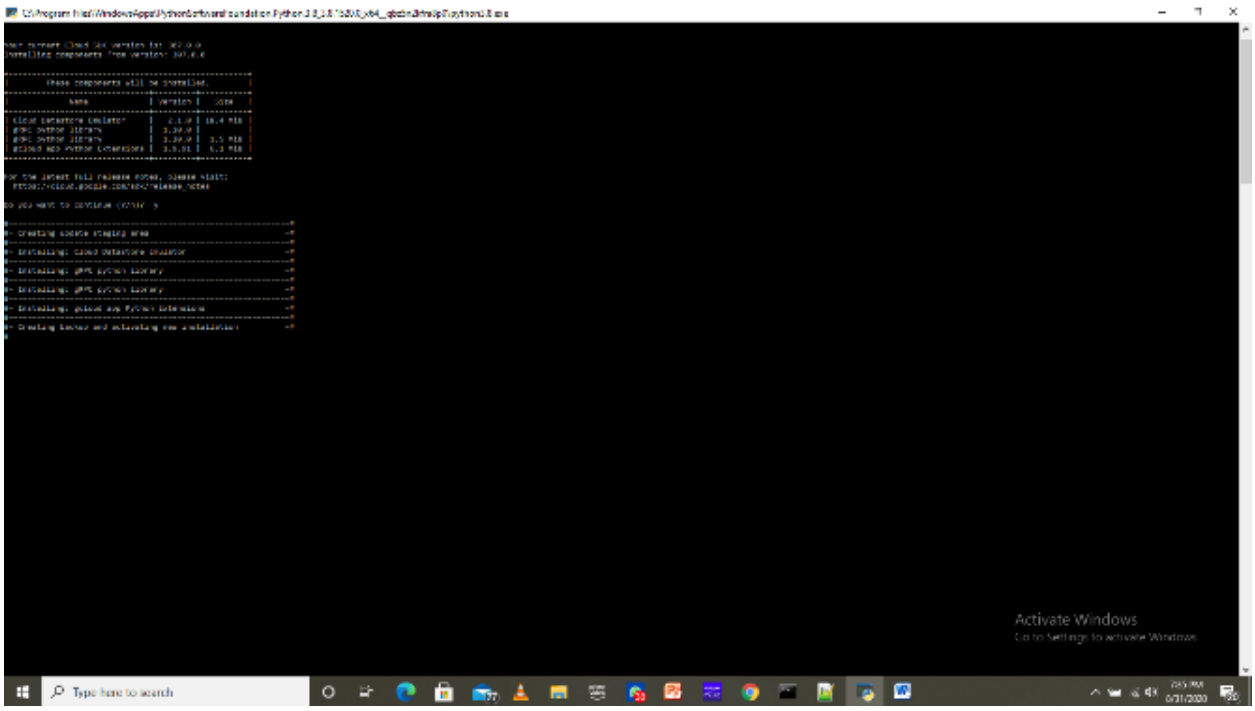
```
runtime: python27
api_version: 1
threadsafe: true
handlers:
- url: /
  script: test.app
```

6. Open google cloud SDK , then type the command dev_appserver.py and specify the location of folder where the above two program files are saved as follows

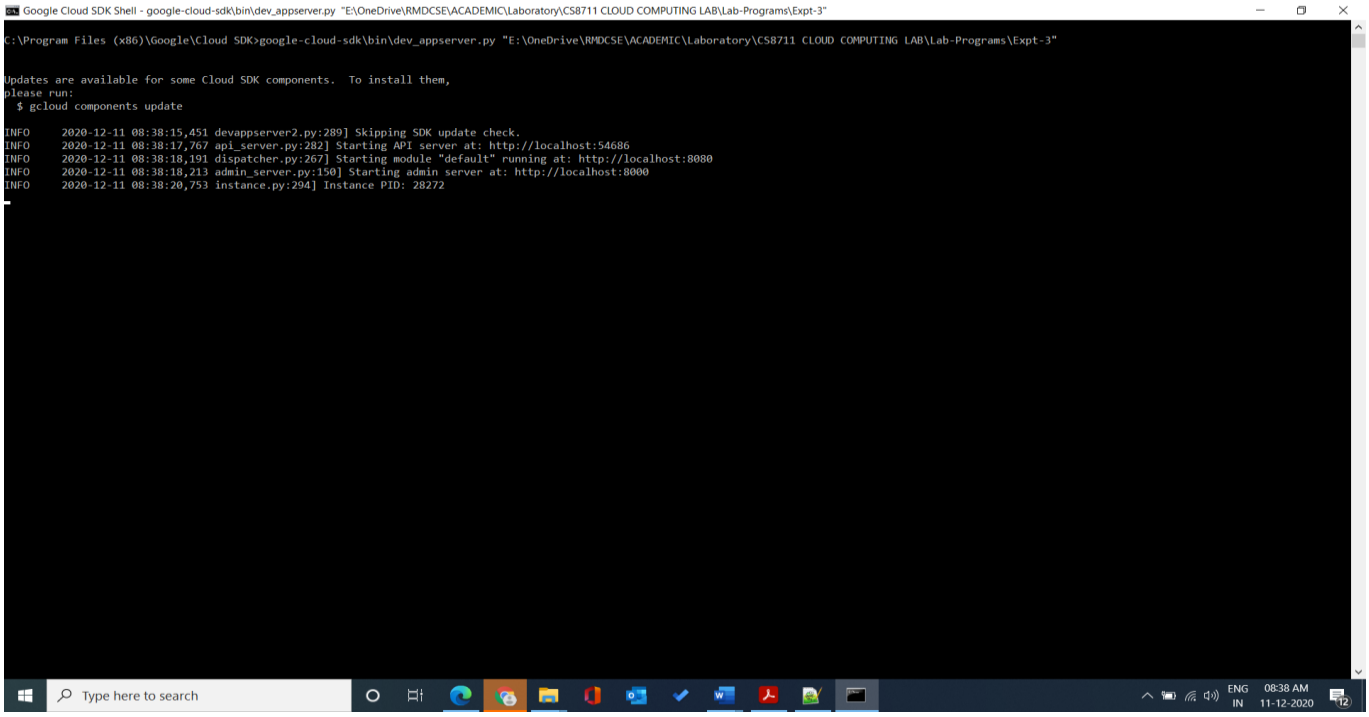
   C:\Program Files (x86)\Google\Cloud SDK>dev_appserver.py

   "E:\OneDrive\RMDCSE\ACADEMIC\Laboratory\CS8711 CLOUD COMPUTING LAB\Lab-Programs\Expt-3"

7. A new window will get open if you are using Python for the first time make sure that you are selecting "Yes" and the press enter

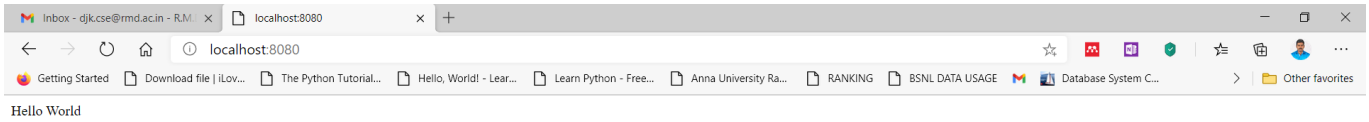



8. On successful execution, you will get output as follows



You can see the Server is running at the local host like : http://localhost:8080

8. Open a web browser and type in the address bar as : http://localhost:8080




**Result:**

Thus, the installation of Google App Engine and Creation of hello world app using python has been done successfully.

| Exp. No: 4 | Use GAE launcher to launch the web applications |
|---|---|
| **Date:** 14/9/2020 | |

**Aim:**

To Use GAE launcher to launch the web applications.

**Procedure:**

1. Install notpad++ editor or any other editor to write program.

2. Download latest version of Python for windows and Install it from

**https://www.python.org/downloads/**

3.Download and install Google App Engine from
**https://cloud.google.com/appengine/downloads?csw=1**
(Download the Cloud SDK installer for windows. )

4. Set the environment variable path to point to python & GAE installation directories as follows
Path= C:\Program Files\Python39\; C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin

5. Write a program for hello world in notpad++ editor as follows ( the indentations in each line of the
code is important) and save it to desired location.

**Program 1: test.py**

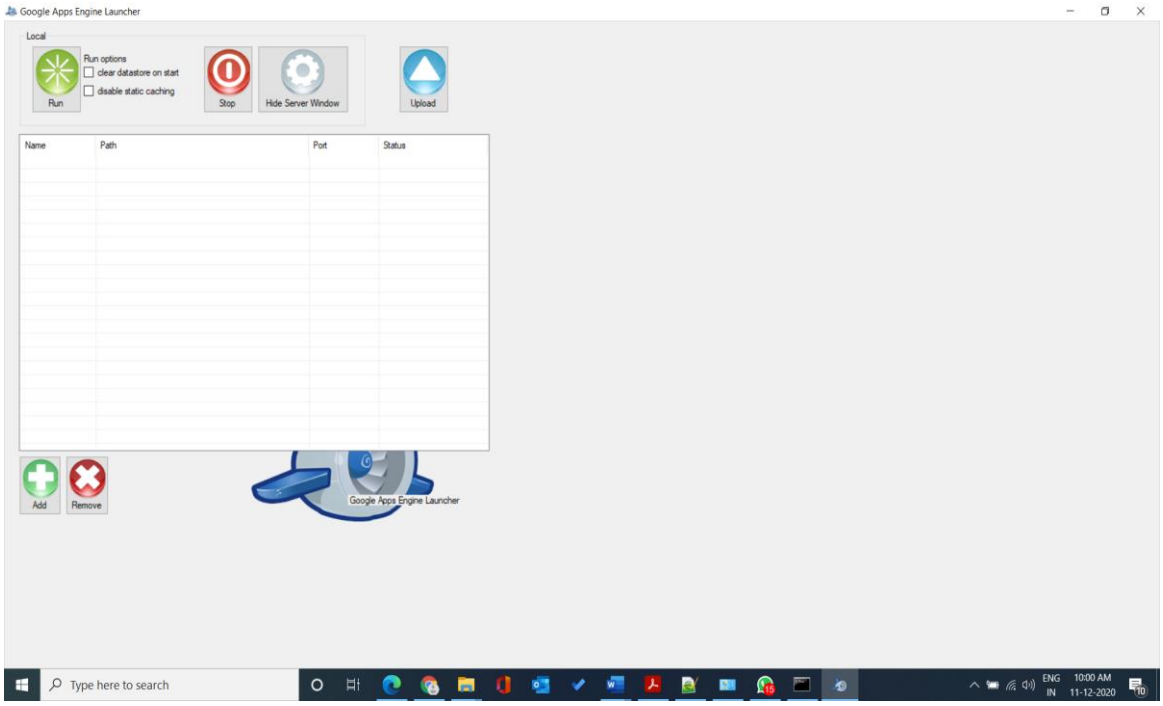
```
import webapp2
class MainPage(webapp2.RequestHandler):
    def get(self):
        self.response.write("Hello World")
app=webapp2.WSGIApplication([('/', MainPage), ], debug=True)
```

**Program 2: app.yaml**

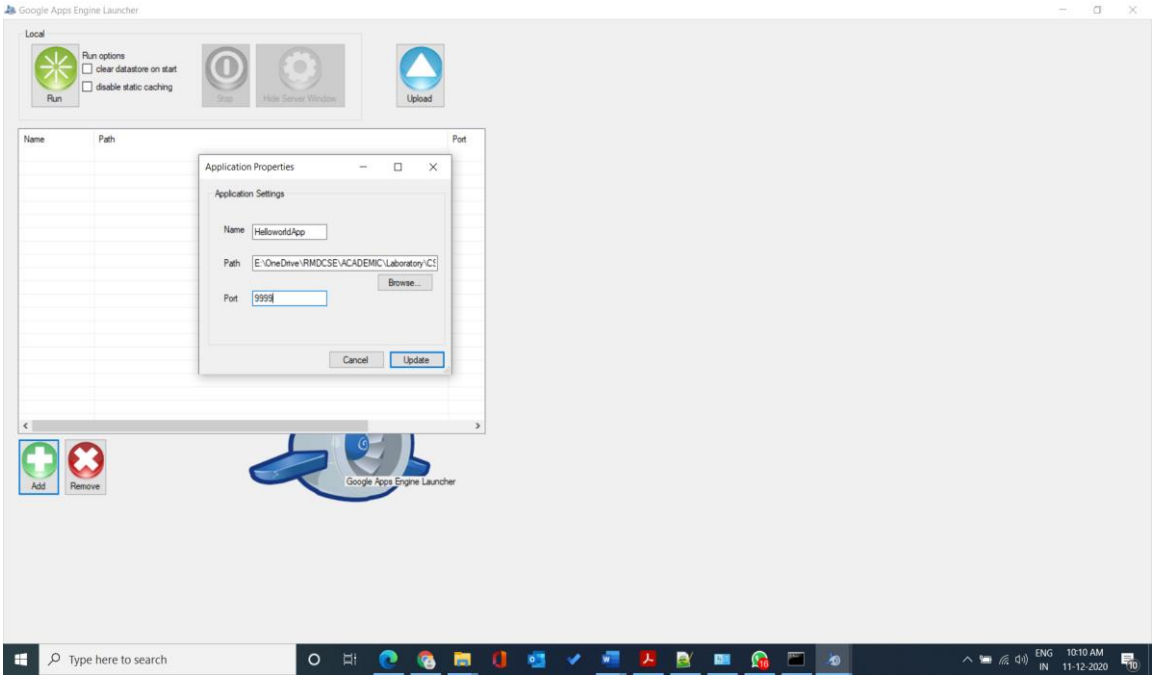
```
runtime: python27
api_version: 1
threadsafe: true
handlers:
- url: /
  script: test.app
```

6. Download and Install the GAE Launcher from
download GAE Launcher (softpedia-secure-download.com)

7. Open the downloaded GAE launcher application



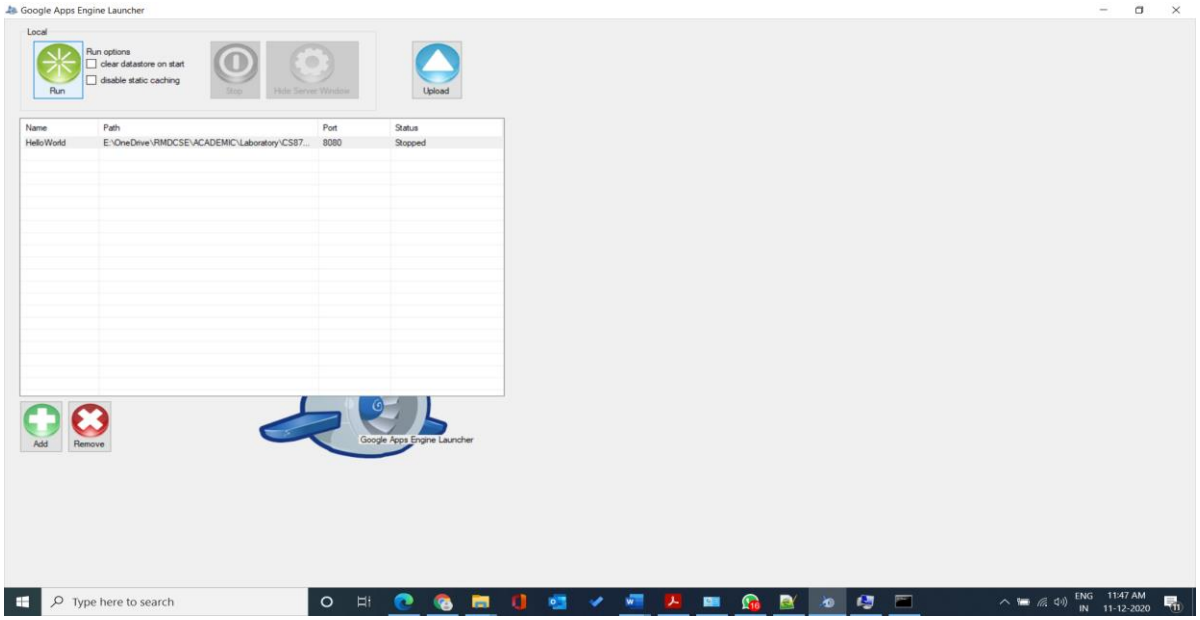

8. Use the Add(+) button to navigate into the directory where you saved your program files



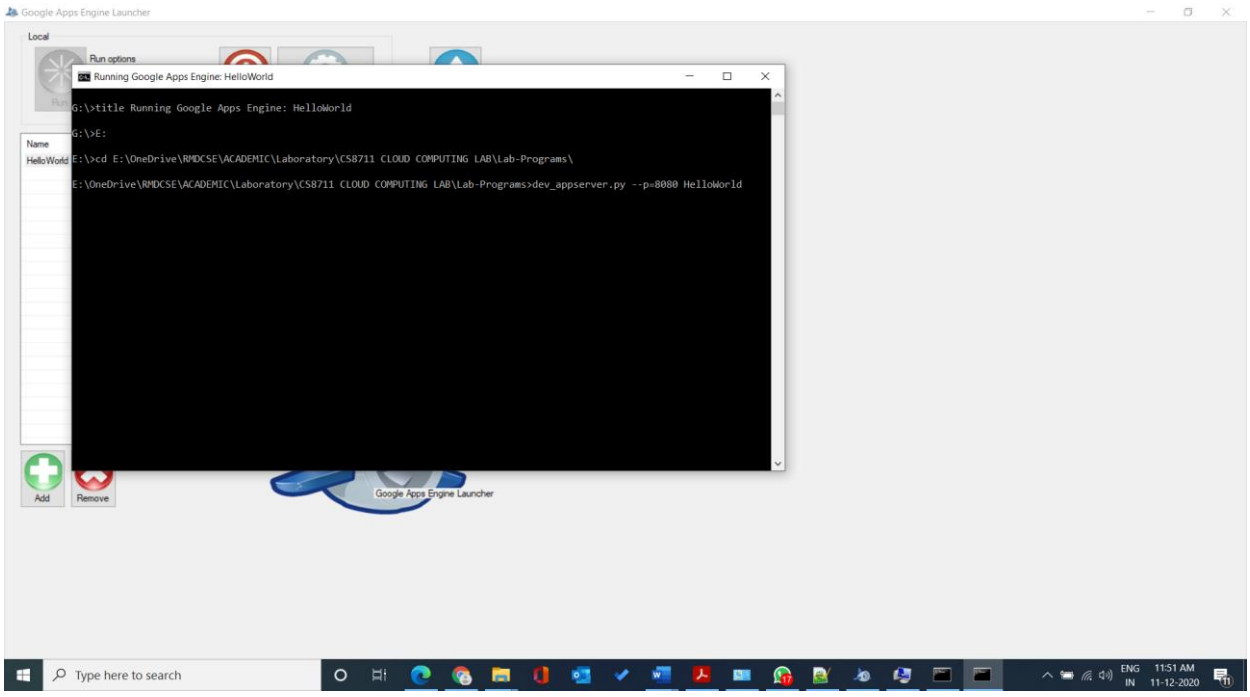

Name: User define name for the app

Path: location of the folder where you saved the program files

9. Select the project and click the run button




10. It will open the command prompt and start running the application as follows




11. On successful execution and if no port conflicts, the application will be running at port number:8080.
12. Open the browser window and Enter **http://localhost:8080.**

    It will display the output “Hello World”

**Result:**

Thus launching of web applications using GAE launcher has been completed successfully.

| Exp. No: 5<br>Date: 21/9/2020 | Simulation of Cloud Scenario using Cloudsim |
|---|---|

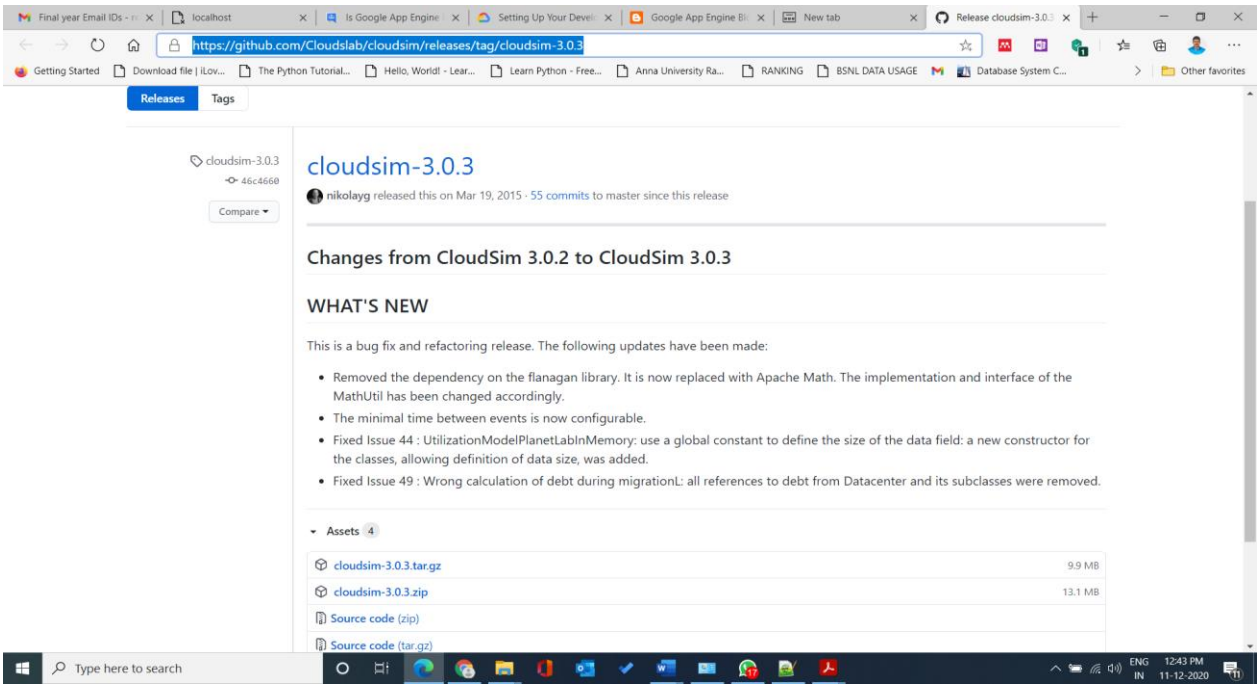
**Aim:**

To simulate the cloud scenario using cloudsim tool and run a scheduling algorithm.

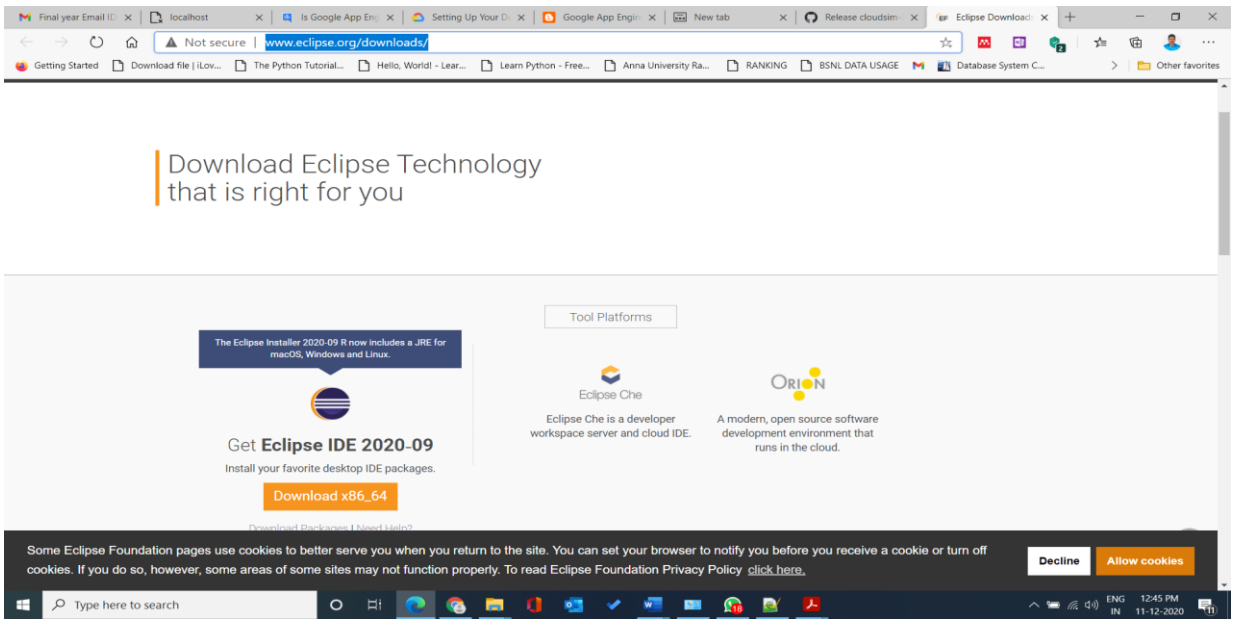
**Procedure:**

**Simulation of cloud scenario:**

1. Go to https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3
2. Download cloudsim-3.0.3.zip in this link

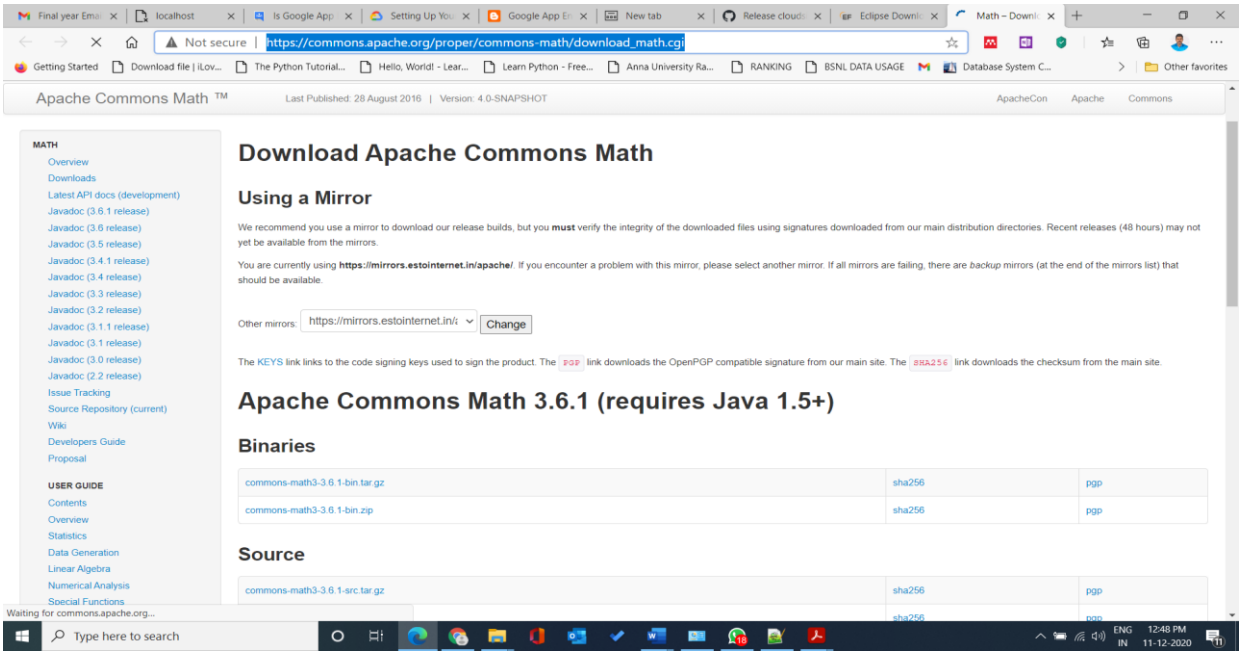



3. Extract the downloaded zip file.
4. Have an eclipse IDE installed.



5. Go to https://commons.apache.org/proper/commons-math/download_math.cgi

6. Download commons-math3-3.6.1-bin.zip from the above link.



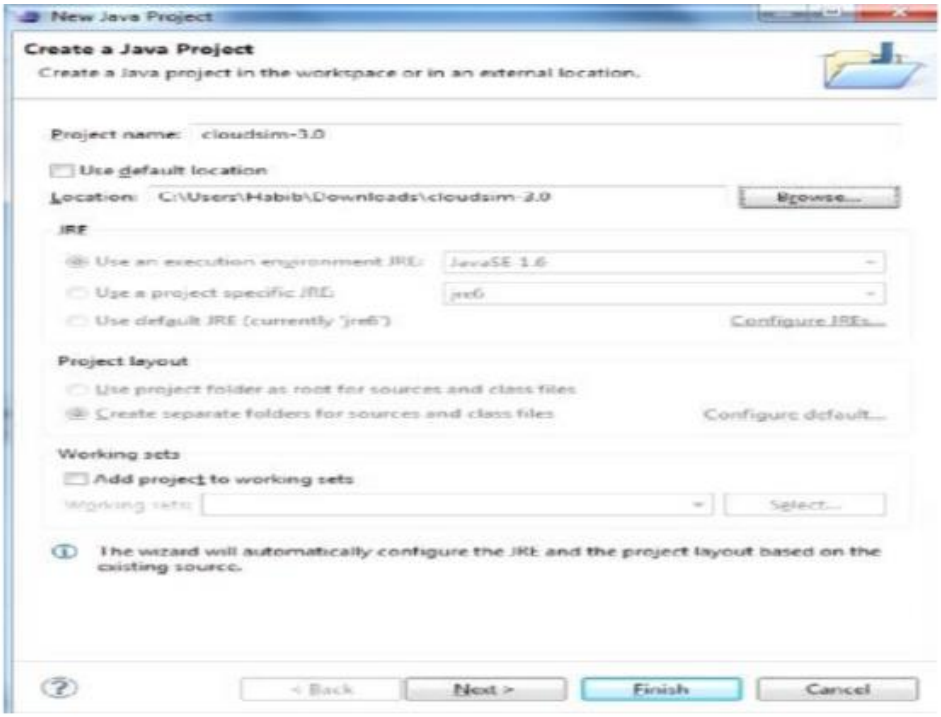

7. Extract the downloaded folder.

8. Open Eclipse SDK and select File--->New--->Java Project.

9. Give a project name( Ex: cloudsim-3.0) and then uncheck the Use Default Location checkbox.

10. Click on browse and select the extracted cloudsim-3.0.3 folder as location.

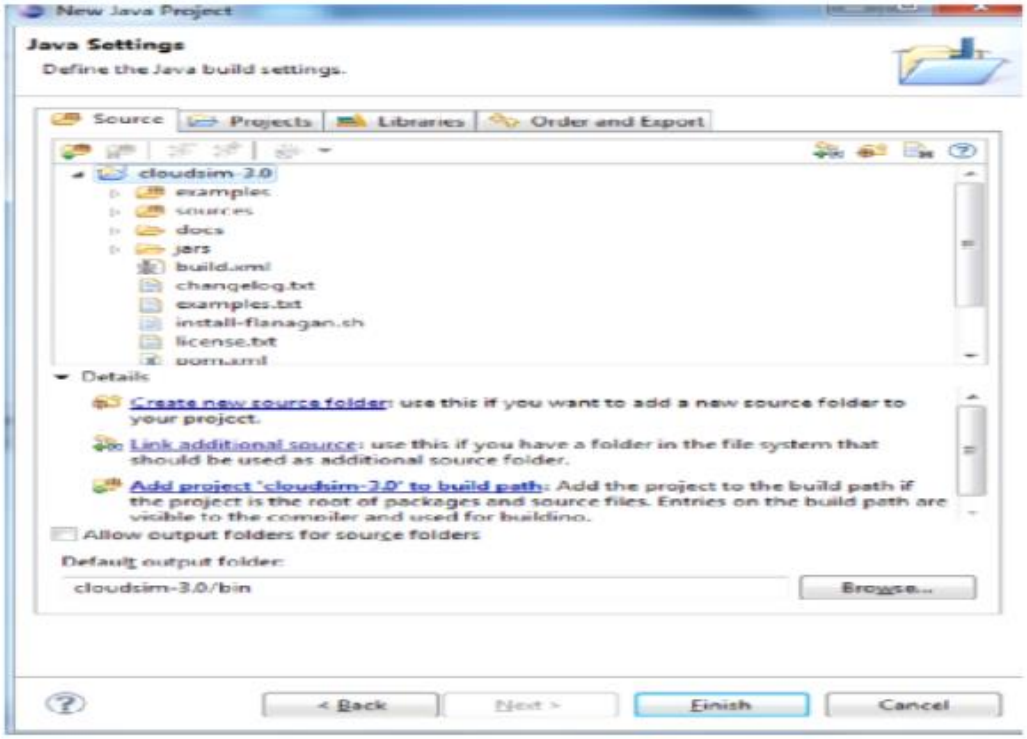
11. Select Next.



12. In the next window select the Libraries tab.

13. Click on add external JARs button and select commons-math3-3.6.1.jar file from the extracted commons-math3-3.6.1 folder.

14. The existing and added JAR files will be displayed in the libraries tab. Check if the selected file has been added and click finish.




15. Now if there are errors in the project, then right click on project name and select properties.

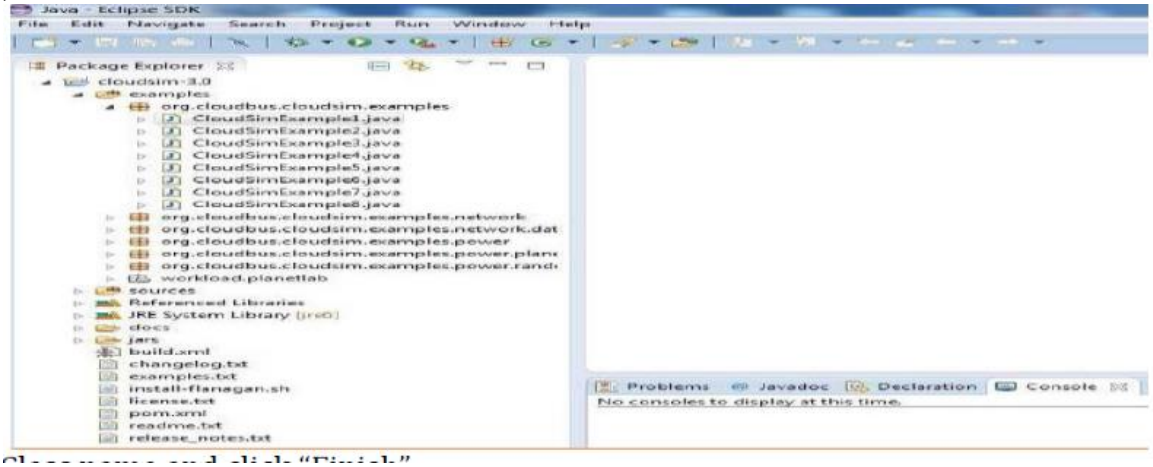
16. Select the java compiler option in the properties window. Enable the project specific settings checkbox. Change the compiler compliance level to 1.7 and allow rebuilding of the project.

17.Simulation Example: ( available in cloudsim package)

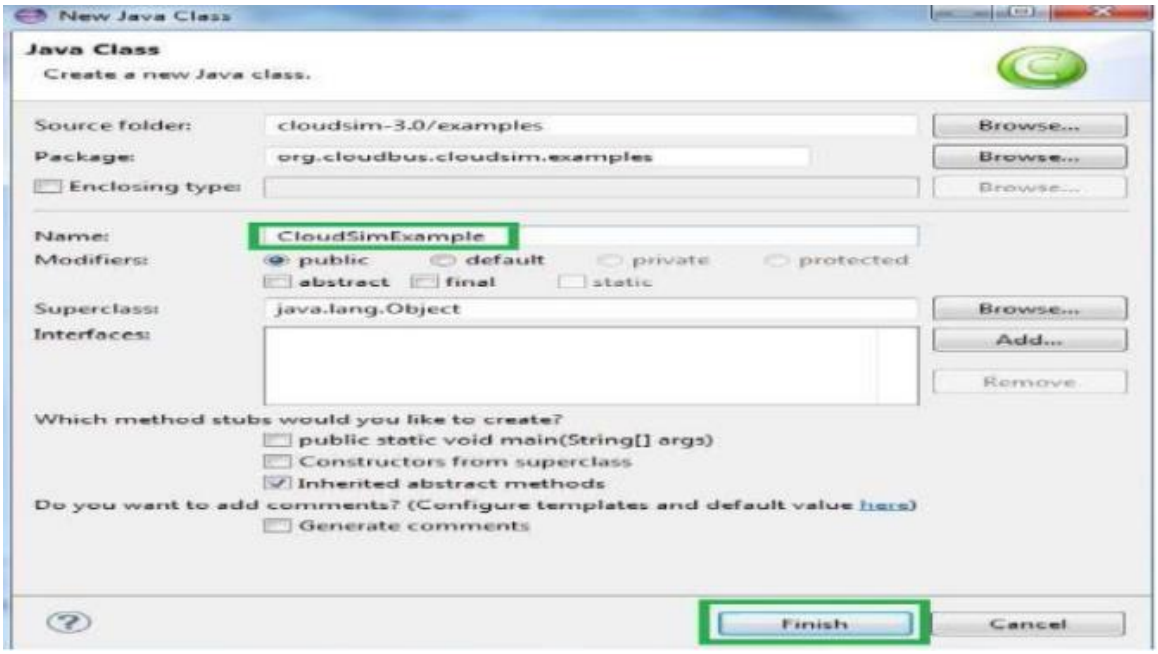
• CloudSimExample1.java : shows how to create a datacenter with one host and run one cloudlet on it.

• CloudSimExample2.java : shows how to create a datacenter with one host and run two cloudlets on it.

18. To create a new class just right click from “org.cloudbus.cloudsim.examples”, select “New” then “Class”




Class name and click “Finish”

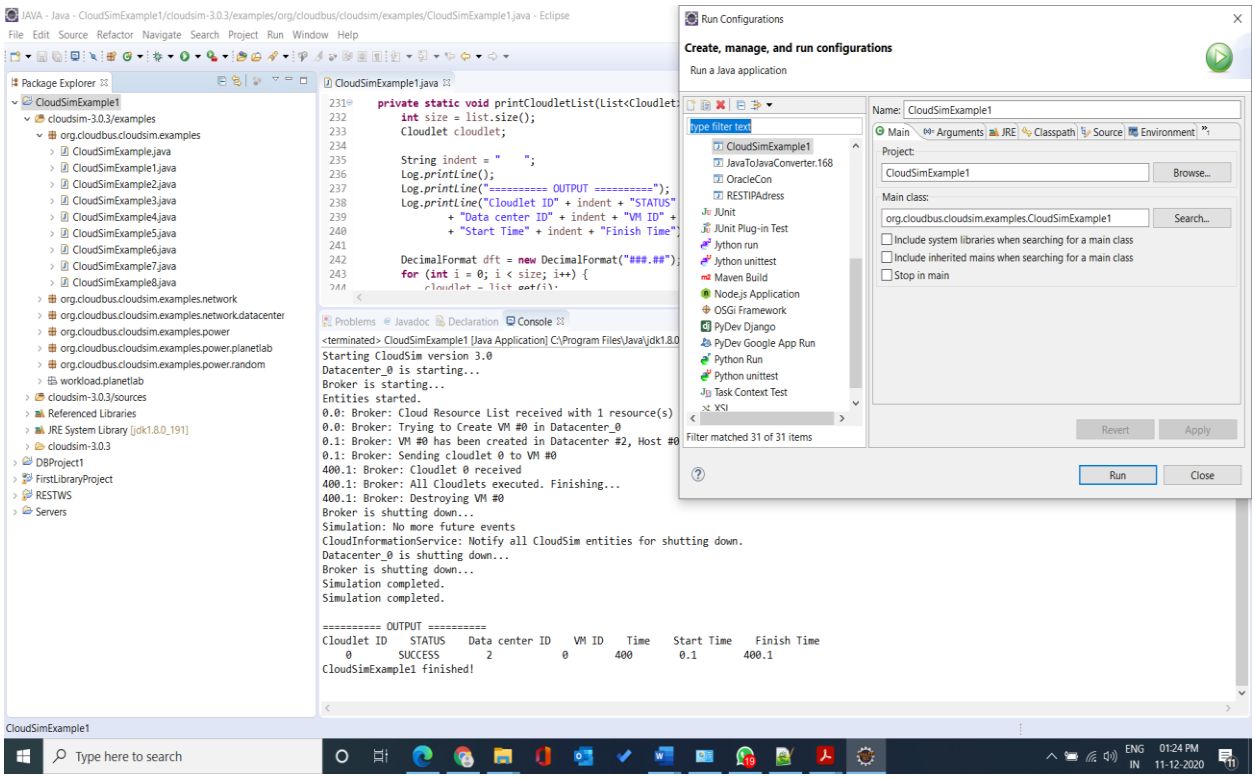



Write the code the application that you want to simulate

19. To run the example, select the project & click run configurations to select the desired class to execute. On

successful execution, the output will be displayed in console window as bellow. Observe the output.

(Running CloudsimExample1 class exist in cloudsim package)



**Simulation of scheduling algorithm in cloudsim:**

The steps to be followed are

1. Initialize the cloudSim package.

2. Create DataCenters to act as resource providers.

3. Create a data center broker. This will help in selecting a data center for usage.

4. Create a list of virtual machines to help in the execution of scheduling algorithm. Submit the list of virtual machines to the broker.

5. Create a list of cloudlet. A cloudlet specifies a set of user requests using an ID and also keeps track of the user to whom the responses has to be sent after processing the request. Submit the list of cloudlets to the broker. Call the required scheduling algorithm using the broker.

6. Now the tasks will get scheduled in the virtual machines.

7. Execute the tasks by starting simulation.

8. Print the results after execution.

Source code:

```
package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
```

```
/**
 * A simple example showing how to create a datacenter with one host and run one
 * cloudlet on it.
 */
public class CloudSimExample1 {

	/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;

	/** The vmlist. */
	private static List<Vm> vmlist;

	/**
	 * Creates main() to run this example.
	 *
	 * @param args the args
	 */
	@SuppressWarnings("unused")
	public static void main(String[] args) {

		Log.printLine("Starting CloudSimExample1...");

		try {
			// First step: Initialize the CloudSim package. It should be called
			// before creating any entities.
			int num_user = 1; // number of cloud users
			Calendar calendar = Calendar.getInstance();
			boolean trace_flag = false; // mean trace events

			// Initialize the CloudSim library
			CloudSim.init(num_user, calendar, trace_flag);

			// Second step: Create Datacenters
			// Datacenters are the resource providers in CloudSim. We need at
			// list one of them to run a CloudSim simulation
			Datacenter datacenter0 = createDatacenter("Datacenter_0");

			// Third step: Create Broker
			DatacenterBroker broker = createBroker();
			int brokerId = broker.getId();

			// Fourth step: Create one virtual machine
			vmlist = new ArrayList<Vm>();

			// VM description
			int vmid = 0;
```

```
int mips = 1000;
long size = 10000; // image size (MB)
int ram = 512; // vm memory (MB)
long bw = 1000;
int pesNumber = 1; // number of cpus
String vmm = "Xen"; // VMM name

// create VM
Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,
new CloudletSchedulerTimeShared());

// add the VM to the vmList
vmlist.add(vm);

// submit vm list to the broker
broker.submitVmList(vmlist);

// Fifth step: Create one Cloudlet
cloudletList = new ArrayList<Cloudlet>();

// Cloudlet properties
int id = 0;
long length = 400000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
cloudlet.setUserId(brokerId);
cloudlet.setVmId(vmid);

// add the cloudlet to the list
cloudletList.add(cloudlet);

// submit cloudlet list to the broker
broker.submitCloudletList(cloudletList);

// Sixth step: Starts the simulation
CloudSim.startSimulation();

CloudSim.stopSimulation();

//Final step: Print results when simulation is over
List<Cloudlet> newList = broker.getCloudletReceivedList();
printCloudletList(newList);
```

```
            Log.printLine("CloudSimExample1 finished!");
        } catch (Exception e) {
            e.printStackTrace();
            Log.printLine("Unwanted errors happen");
        }
    }

    /**
     * Creates the datacenter.
     *
     * @param name the name
     *
     * @return the datacenter
     */
    private static Datacenter createDatacenter(String name) {

        // Here are the steps needed to create a PowerDatacenter:
        // 1. We need to create a list to store
        // our machine
        List<Host> hostList = new ArrayList<Host>();

        // 2. A Machine contains one or more PEs or CPUs/Cores.
        // In this example, it will have only one core.
        List<Pe> peList = new ArrayList<Pe>();

        int mips = 1000;

        // 3. Create PEs and add these into a list.
        peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and
MIPS
                                                                    Rating

        // 4. Create Host with its id and list of PEs and add them to the list
        // of machines
        int hostId = 0;
        int ram = 2048; // host memory (MB)
        long storage = 1000000; // host storage
        int bw = 10000;

        hostList.add(
            new Host(
                hostId,
                new RamProvisionerSimple(ram),
                new BwProvisionerSimple(bw),
                storage,
                peList,
                new VmSchedulerTimeShared(peList)
```

```
                        )
                );// This is our machine

                // 5. Create a DatacenterCharacteristics object that stores the
                // properties of a data center: architecture, OS, list of
                // Machines, allocation policy: time- or space-shared, time zone
                // and its price (G$/Pe time unit).
                String arch = "x86"; // system architecture
                String os = "Linux"; // operating system
                String vmm = "Xen";
                double time_zone = 10.0; // time zone this resource located
                double cost = 3.0; // the cost of using processing in this resource
                double costPerMem = 0.05; // the cost of using memory in this resource
                double costPerStorage = 0.001; // the cost of using storage in this
                                                                        // resource
                double costPerBw = 0.0; // the cost of using bw in this resource
                LinkedList<Storage> storageList = new LinkedList<Storage>(); // we are not
adding SAN

        // devices by now

                DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                                arch, os, vmm, hostList, time_zone, cost, costPerMem,
                                costPerStorage, costPerBw);

                // 6. Finally, we need to create a PowerDatacenter object.
                Datacenter datacenter = null;
                try {
                        datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
                } catch (Exception e) {
                        e.printStackTrace();
                }

                return datacenter;
        }

        // We strongly encourage users to develop their own broker policies, to
        // submit vms and cloudlets according
        // to the specific rules of the simulated scenario
        /**
         * Creates the broker.
         *
         * @return the datacenter broker
         */
        private static DatacenterBroker createBroker() {
                DatacenterBroker broker = null;
```

```
            try {
                broker = new DatacenterBroker("Broker");
            } catch (Exception e) {
                e.printStackTrace();
                return null;
            }
            return broker;
        }

        /**
         * Prints the Cloudlet objects.
         *
         * @param list list of Cloudlets
         */
        private static void printCloudletList(List<Cloudlet> list) {
            int size = list.size();
            Cloudlet cloudlet;

            String indent = "    ";
            Log.printLine();
            Log.printLine("========== OUTPUT ==========");
            Log.printLine("Cloudlet ID" + indent + "STATUS" + indent
                    + "Data center ID" + indent + "VM ID" + indent + "Time" + indent
                    + "Start Time" + indent + "Finish Time");

            DecimalFormat dft = new DecimalFormat("###.##");
            for (int i = 0; i < size; i++) {
                cloudlet = list.get(i);
                Log.print(indent + cloudlet.getCloudletId() + indent + indent);

                if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                    Log.print("SUCCESS");

                    Log.printLine(indent + indent + cloudlet.getResourceId()
                            + indent + indent + indent + cloudlet.getVmId()
                            + indent + indent
                            + dft.format(cloudlet.getActualCPUTime()) + indent
                            + indent + dft.format(cloudlet.getExecStartTime())
                            + indent + indent
                            + dft.format(cloudlet.getFinishTime()));
                }
            }
        }

}
```

**Output:**

```
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
   0        SUCCESS        2            0        400        0.1        400.1
```

**Result:**

Thus the simulation of cloud scenario using cloudsim has been completed successfully.

| Exp. No: 6 | Transfer the files from one Virtual Machine to another Virtual Machine |
|---|---|
| **Date:** 28/9/2020 | |

**Aim:**

To find the procedure to transfer the files from one virtual machine to another Virtual Machine.

**Procedure:**

***Prerequisite:***

Install two instance of the virtual machine using Virtualbox and follow the methods given below for transferring files between virtual machines.

***Steps***

1. You can copy few (or more) lines with copy & paste mechanism.
   For this you need to share clipboard between host OS and guest OS, installing Guest Addition on both the virtual machines (probably setting bidirectional and restarting them).
   You copy from guest OS in the clipboard that is shared with the host OS.
   Then you paste from the host OS to the second guest OS.
2. You can enable drag and drop too with the same method (Click on the machine, settings, general, advanced, drag and drop set to bidirectional)
3. You can have common Shared Folders on both virtual machines and use one of the directory shared as buffer to copy. Installing Guest Additions, you have the possibility to set Shared Folders too. As you put a file in a shared folder from host OS or from guest OS, is immediately visible to the other. (Keep in mind that can arise some problems for date/time of the files when there are different clock settings on the different virtual machines).If you use the same folder shared on more machines you can exchange files directly copying them in this folder.
4. You can use usual method to copy files between 2 different computers with client-server application. (e.g. scp with sshd active for linux, winscp... you can get some info about SSH servers e.g. here). You need an active server (sshd) on the receiving machine and a client on the sending machine. Of course, you need to have the authorization setted (via password or, better, via an automatic authentication method).
   **Note:** many Linux/Ubuntu distribution install sshd by default: you can see if it is running with pgrep sshd from a shell. You can install with sudo apt-get install openssh-server.

5. You can mount part of the file system of a virtual machine via NFS or SSHFS on the other, or you can share file and directory with Samba.

You should remember that you are dialling with a little network of machines with different operative systems, and in particular:

- Each virtual machine has its own operative system running on and acts as a physical machine.
- Each virtual machine is an instance of a program owned by an user in the hosting operative system and should undergo the restrictions of the user in the hosting OS.

  E.g Let we say that Hastur and Meow are users of the hosting machine, but they did not allow each other to see their directories (no read/write/execute authorization). When each of them run a virtual machine, for the hosting OS those virtual machine are two normal programs owned by Hastur and Meow and cannot see the private directory of the other user.This is a restriction due to the hosting OS. It's easy to overcame it: it's enough to give authorization to read/write/execute to a directory or to chose a different directory in which both users can read/write/execute.
- Windows likes mouse and Linux fingers. :-)

  I mean I suggest you to enable Drag & drop to be cosy with the Windows machines and the Shared folders or to be cosy with Linux. When you will need to be fast with Linux you will feel the need of ssh-keygen and to Generate once SSH Keys to copy files on/from a remote machine without writing password anymore. In this way it functions bash auto-completion remotely too!

**Output:**

The screenshots of file sharing process between the Virtual Machine

**Result:**

Thus, the procedure for file transfer between virtual machines id tested successfully.

| Exp. No: 7 | Find a procedure to launch virtual machine using trystack |
|---|---|
| **Date:** 12/10/2020 | |

**Aim:**

To find a procedure to launch virtual machine using trystack (Openstack demo version).

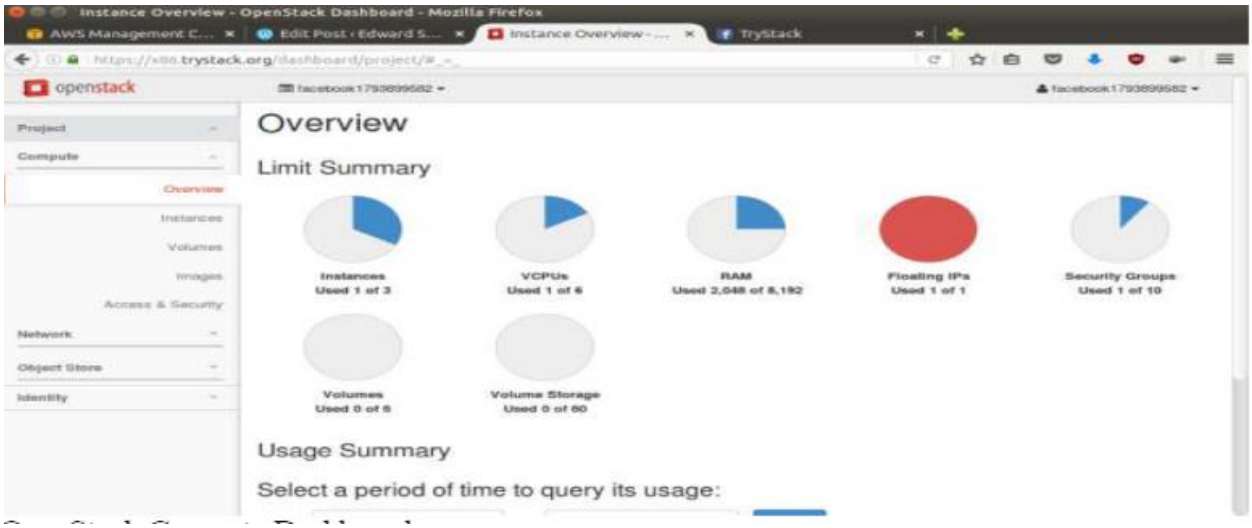
**Procedure:**

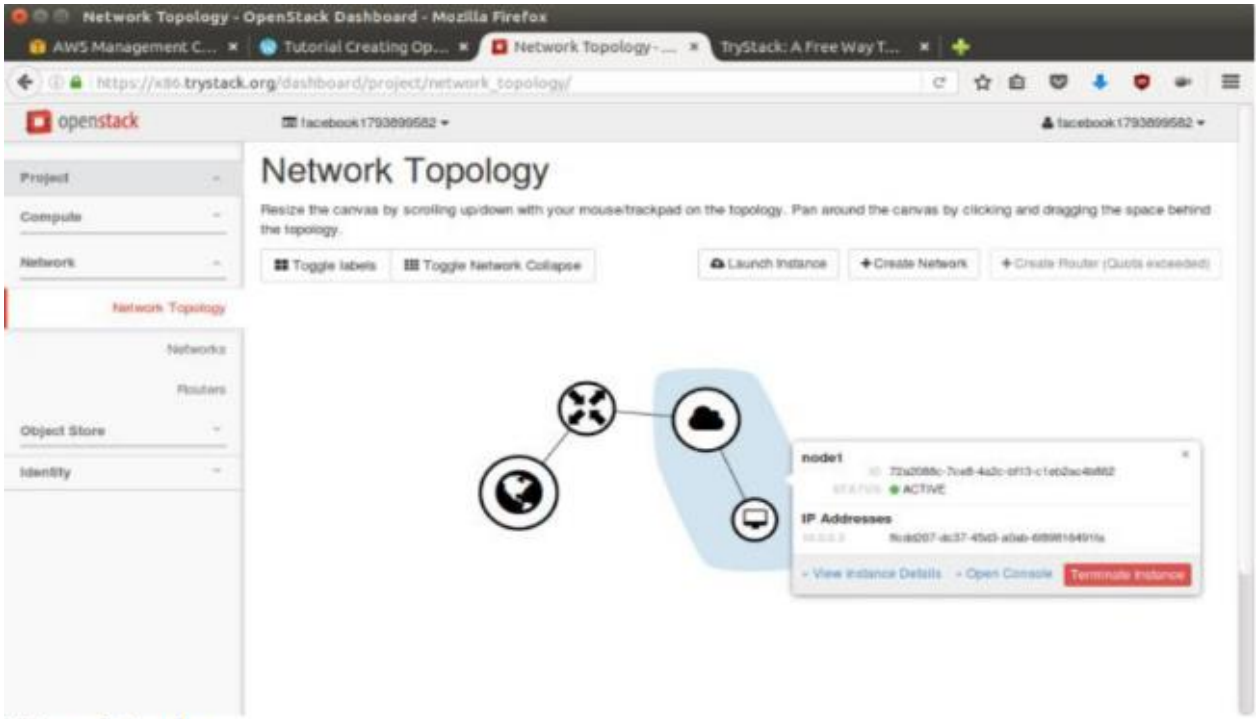
**OpenStack** is an open-source software cloud computing platform. OpenStack is primarily used for deploying an infrastructure as a service (IaaS) solution like Amazon Web Service (AWS). In other words, you can *make your own AWS* by using OpenStack. If you want to try out OpenStack, **TryStack** is the easiest and free way to do it.

In order to try OpenStack in TryStack, you must register yourself by joining TryStack Facebook Group. The acceptance of group needs a couple days because it's approved manually. After you have been accepted in the TryStack Group, you can log in TryStack.


TryStack.org Homepage

I assume that you already join to the Facebook Group and login to the dashboard. After you log in to the TryStack, you will see the Compute Dashboard like:






Network topology

As you see from the image above, the instance will be connected to a local network and the local network will be connected to internet.

**Step 1: Create Network**

Network? Yes, the network in here is our own local network. So, your instances will be not mixed up with the others. You can imagine this as your own LAN (Local Area Network) in the cloud.

1. Go to **Network > Networks** and then click **Create Network**.
2. In **Network** tab, fill **Network Name** for example internal and then click **Next**.
3. In **Subnet** tab,
1. Fill **Network Address** with appropriate CIDR, for example 192.168.1.0/24. Use private network CIDR block as the best practice.
2. Select **IP Version** with appropriate IP version, in this case IPv4.
3. Click **Next**.
4. In **Subnet Details** tab, fill **DNS Name Servers** with 8.8.8.8 (Google DNS) and then click **Create**.

**Step 2: Create Instance**

Now, we will create an instance. The instance is a virtual machine in the cloud, like AWS EC2. You need the instance to connect to the network that we just created in the previous step.

1. Go to **Compute > Instances** and then click **Launch Instance**.
2. In **Details** tab,
    1. Fill **Instance Name**, for example Ubuntu 1.
    2. Select **Flavor**, for example m1.medium.
    3. Fill **Instance Count** with **1**.
    4. Select **Instance Boot Source** with **Boot from Image**.
    5. Select **Image Name** with **Ubuntu 14.04 amd64 (243.7 MB)** if you want install Ubuntu 14.04 in your virtual machine.

3. In **Access & Security** tab,
    1. Click [+] button of **Key Pair** to import key pair. This key pair is a public and private key that we will use to connect to the instance from our machine.
    2. In **Import Key Pair** dialog,
        1. Fill **Key Pair Name** with your machine name (for example Edward-Key).
        2. Fill **Public Key** with your **SSH public key** (usually is in ~/.ssh/id_rsa.pub). See description in Import Key Pair dialog box for more information. If you are using Windows, you can use **Puttygen** to generate key pair.
        3. Click **Import key pair**.
    3. In **Security Groups**, mark/check **default**.
    4. In **Networking** tab,
        1. In **Selected Networks**, select network that have been created in Step 1, for example internal.
    5. Click **Launch**.
    6. If you want to create multiple instances, you can repeat step 1-5. I created one more instance with instance name Ubuntu 2.

**Step 3: Create Router**

I guess you already know what router is. In the step 1, we created our network, but it is isolated. It doesn't connect to the internet. To make our network has an internet connection, we need a router that running as the gateway to the internet.

1. Go to **Network > Routers** and then click **Create Router**.
2. Fill **Router Name** for example router1 and then click **Create router**.
3. Click on your **router name link**, for example router1, **Router Details** page.
4. Click **Set Gateway** button in upper right:
    1. Select **External networks** with **external**.
    2. Then **OK**.
5. Click **Add Interface** button.
    1. Select **Subnet** with the network that you have been created in Step 1.
    2. Click **Add interface**.
6. Go to **Network > Network Topology**. You will see the network topology. In the example, there are two network, i.e. external and internal, those are bridged by a router. There are instances those are joined to internal network.

**Step 4: Configure Floating IP Address**

*Floating IP address* is public IP address. It makes your instance is accessible from the internet. When you launch your instance, the instance will have a private network IP, but no public IP. In OpenStack, the public IPs is collected in a pool and managed by admin (in our case is TryStack). You need to request a public (floating) IP address to be assigned to your instance.

1. Go to **Compute > Instance**.
2. In one of your instances, click **More > Associate Floating IP**.
3. In **IP Address**, click Plus [+].
4. Select **Pool** to **external** and then click **Allocate IP**.
5. Click **Associate**.
6. Now you will get a public IP, e.g. 8.21.28.120, for your instance.

**Step 5: Configure Access & Security**

OpenStack has a feature like a firewall. It can whitelist/blacklist your in/out connection. It is called *Security Group*.

1. Go to **Compute > Access & Security** and then open **Security Groups** tab.
2. In **default** row, click **Manage Rules**.
3. Click **Add Rule**, choose **ALL ICMP** rule to enable ping into your instance, and then click **Add**.
4. Click **Add Rule**, choose **HTTP** rule to open HTTP port (port 80), and then click **Add**.

5. Click **Add Rule**, choose **SSH** rule to open SSH port (port 22), and then click **Add**.
6. You can open other ports by creating new rules.

**Step 6: SSH to Your Instance**

Now, you can SSH your instances to the floating IP address that you got in the step 4. If you are using Ubuntu image, the SSH user will be ubuntu.

**Result:**

Thus, the virtual machine is launched by using trystack by following the procedure successfully.

| Exp. No: 8<br>Date: 19/10/2020 | Install Hadoop single node cluster and run word count program |
|---|---|

**Aim:**

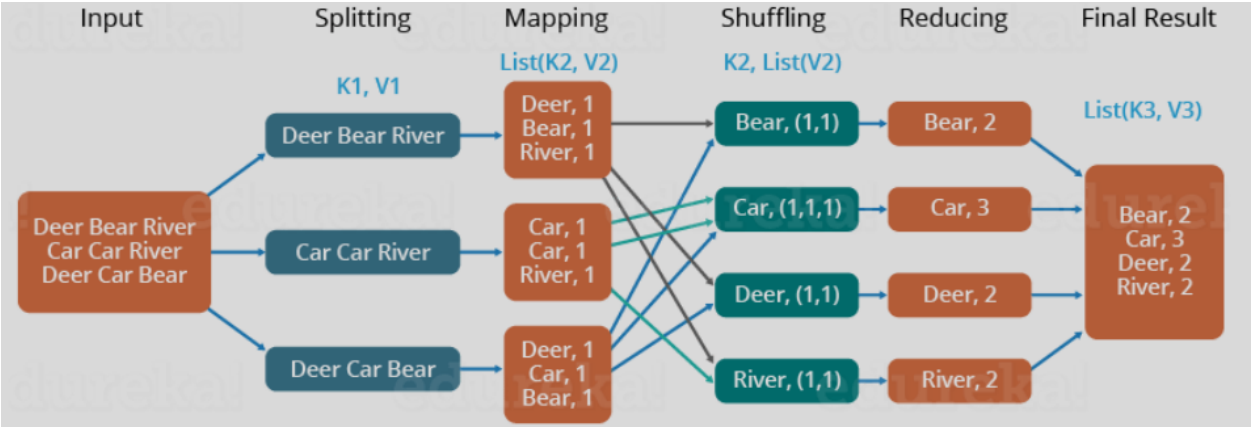
To Install Hadoop single node cluster and run word count program using Mapreduce algorithm.

**Procedure:**

## MapReduce Word Count

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

**Dear, Bear, River, Car, Car, River, Deer, Car and Bear**




## Pre-requisite

**Step-A : SSH Installation –** Installation of SSH

**Step-B : Java Installation –** Installation of Java

**Step-C : Hadoop Installation –** Installation of Hadoop

## A) SSH Installation

SSH is used to interact with the master and slaves computer without any prompt for password. First of all create a Hadoop user on the master and slave systems

- **# useradd hadoop**
- **# passwd Hadoop**

To map the nodes open the hosts file present in /etc/ folder on all the machines and put the ip address along with their host name.

- **# vi /etc/hosts**

Enter the lines below

- **190.12.1.114 hadoop-master**
- **190.12.1.121 hadoop-salve-one**
- **190.12.1.143 hadoop-slave-two**

Set up SSH key in every node so that they can communicate among themselves without password. Commands for the same are:

- **# su hadoop**
- **$ ssh-keygen -t rsa**
- **$ ssh-copy-id -i ~/.ssh/id_rsa.pub**
- **$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp1@hadoop-slave-1**
    - **$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp2@hadoop-slave-2**
    - **$ chmod 0600 ~/.ssh/authorized_keys**
    - **$ exit**

## **B**) Java Installation

1) Type "java -version" in prompt to find if the java is installed or not. If not then download java from http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html . The tar filejdk-7u71-linux-x64.tar.gz will be downloaded to your system.

2) Extract the file using the below command

- **#tar zxf jdk-7u71-linux-x64.tar.gz**

3) To make java available for all the users of UNIX move the file to /usr/local and set the path. In the prompt switch to root user and then type the command below to move the jdk to /usr/lib using below command

- **# mv jdk1.7.0_71 /usr/lib/**

Now in ~/.bashrc file add the following commands to set up the path.

- **# export JAVA_HOME=/usr/lib/jdk1.7.0_71**
- **# export PATH=PATH:$JAVA_HOME/bin**

## C) Hadoop Installation

Hadoop can be downloaded from http://developer.yahoo.com/hadoop/tutorial/module3.html

Now extract the Hadoop and copy it to a location.

- **$ mkdir /usr/hadoop**
- **$ sudo tar vxzf hadoop-2.2.0.tar.gz ?c /usr/hadoop**

Change the ownership of Hadoop folder

- **$ sudo chown -R hadoop  usr/hadoop**

Change the Hadoop configuration files:

All the files are present in **/usr/local/Hadoop/etc/hadoop**

1) In hadoop-env.sh file add

- **export JAVA_HOME=/usr/lib/jvm/jdk/jdk1.7.0_71**

2) **In core-site.xml add following between configuration tabs,**

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://hadoop-master:9000</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
</configuration>
```

3) **In hdfs-site.xml add following between configuration tabs**,

```
<configuration>
<property>
<name>dfs.data.dir</name>
<value>usr/hadoop/dfs/name/data</value>
<final>true</final>
</property>
<property>
<name>dfs.name.dir</name>
<value>usr/hadoop/dfs/name</value>
<final>true</final>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
```

4) **Open the Mapred-site.xml and make the change as shown below**

```
<configuration>
<property>
```

```
<name>mapred.job.tracker</name>
<value>hadoop-master:9001</value>
</property>
</configuration>
```

5) **Finally, update your $HOME/.bahsrc**

```
cd $HOME
vi .bashrc
Append following lines in the end and save and exit
#Hadoop variables
export JAVA_HOME=/usr/lib/jvm/jdk/jdk1.7.0_71
export HADOOP_INSTALL=/usr/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
```

**On the slave machine install Hadoop using the command below**

```
# su hadoop
$ cd /opt/hadoop
$ scp -r hadoop hadoop-slave-one:/usr/hadoop
$ scp -r hadoop hadoop-slave-two:/usr/Hadoop
```

**Configure master node and slave node**

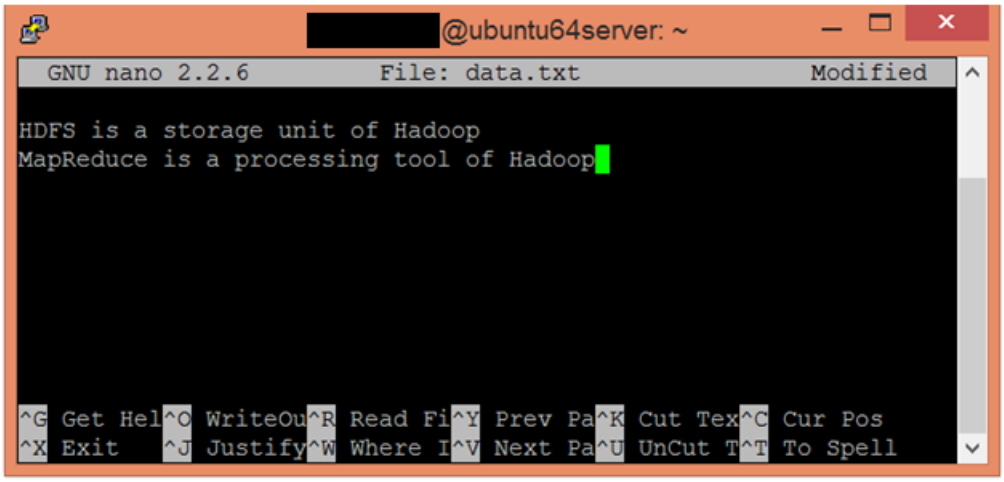
```
$ vi etc/hadoop/masters
hadoop-master
$ vi etc/hadoop/slaves
hadoop-slave-one
hadoop-slave-two
```

**After this format the name node and start all the deamons**

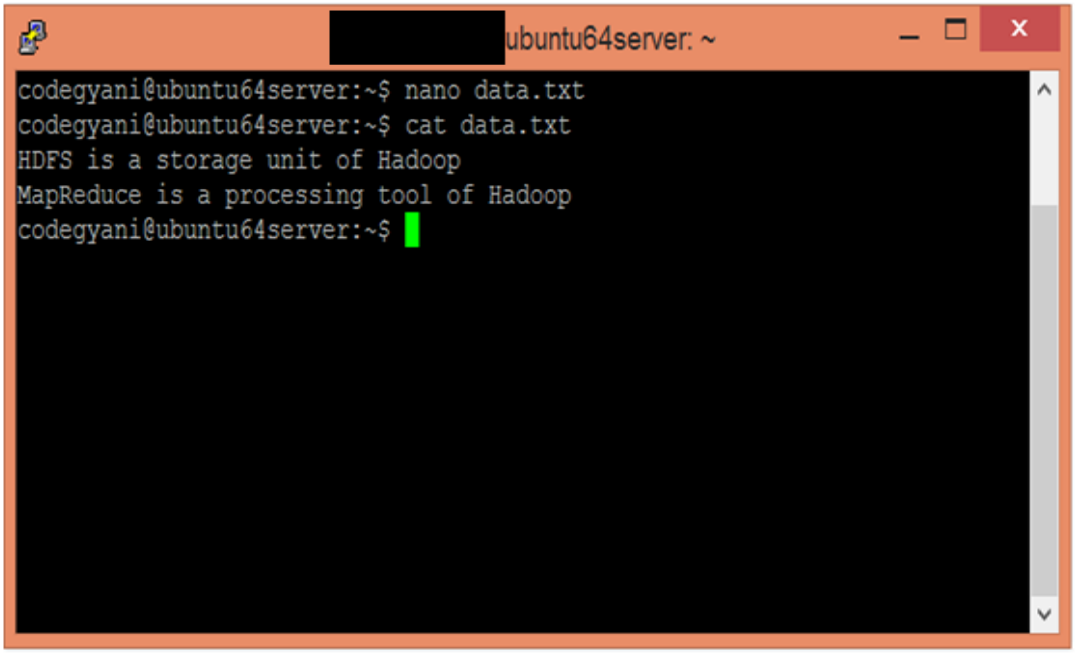
```
# su hadoop
$ cd /usr/hadoop
$ bin/hadoop namenode -format
$ cd $HADOOP_HOME/sbin
$ start-all.sh
```

Steps to execute MapReduce word count example

**Step-1 :** Create a text file in your local machine and write some text into it.
$ nano data.txt

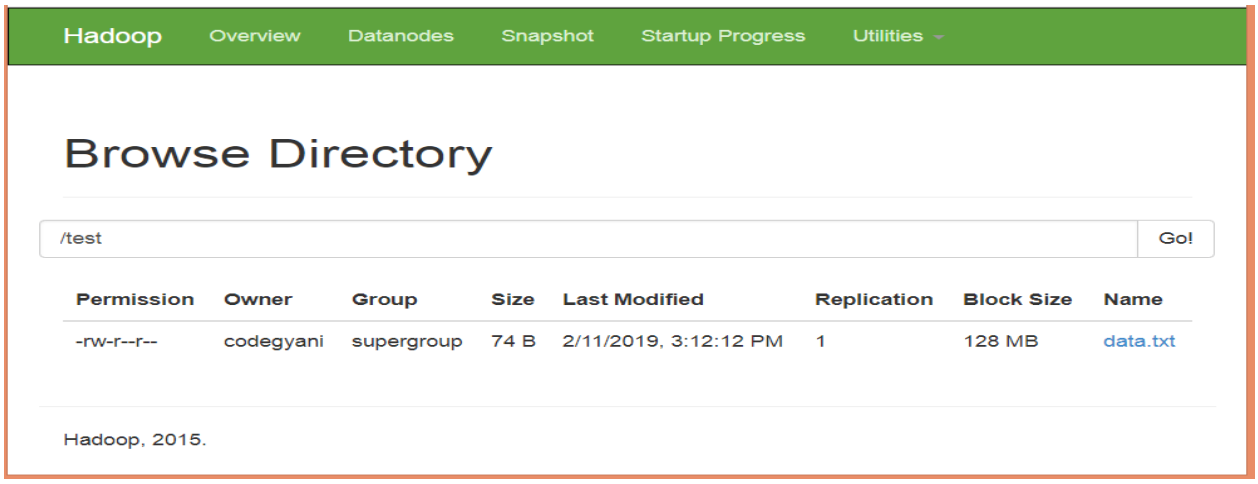



**Step-2 :** Check the text written in the data.txt file.
$ cat data.txt




**Step-3 :** Create a directory in HDFS, where to kept text file.
$ hdfs dfs -mkdir /test

**Step-4 :** Upload the data.txt file on HDFS in the specific directory.
$ hdfs dfs -put /home/code/data.txt /test




**Step-5 :** Write the MapReduce program using eclipse.

**WC_Mapper.java**

```
package com.javatpoint;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,IntWritable>
{
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,
         Reporter reporter) throws IOException{
      String line = value.toString();
      StringTokenizer  tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()){
         word.set(tokenizer.nextToken());
         output.collect(word, one);
      }
   }
}
```

**WC_Reducer.java**

```
package com.javatpoint;
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
```

```
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WC_Reducer  extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable>
{
public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
 Reporter reporter) throws IOException {
int sum=0;
while (values.hasNext()) {
sum+=values.next().get();
}
output.collect(key,new IntWritable(sum));
}
}
```

**WC_Runner.java**

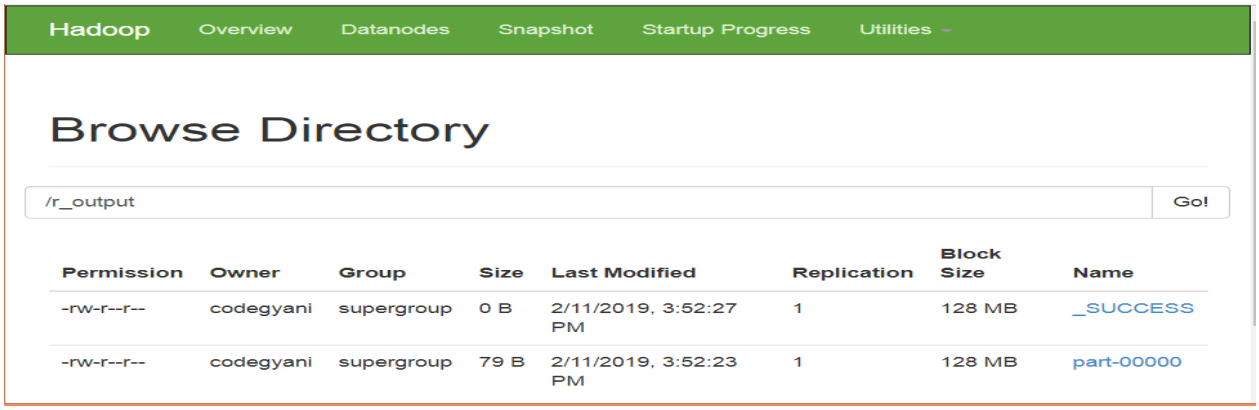
```
package com.javatpoint;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
public class WC_Runner {
  public static void main(String[] args) throws IOException{
    JobConf conf = new JobConf(WC_Runner.class);
    conf.setJobName("WordCount");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(WC_Mapper.class);
    conf.setCombinerClass(WC_Reducer.class);
    conf.setReducerClass(WC_Reducer.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf,new Path(args[0]));
    FileOutputFormat.setOutputPath(conf,new Path(args[1]));
    JobClient.runJob(conf);
  }
}
```

**Step-6 :** Create the jar file of this program and name it **countworddemo.jar.**

**Step-7 :** Run the jar file

hadoop jar /home/codegyani/wordcountdemo.jar com.javatpoint.WC_Runner /test/data.txt /r_output

**Step-8 :** The output is stored in /r_output/part-00000




**Step-9 :** Now execute the command to see the output.
hdfs dfs -cat /r_output/part-00000


**Result:**

Thus the installation of Hadoop one node cluster has been completed successfully.